;login:

THE MAGAZINE OF USENIX & SAGE

June 2000 • volume 25 • number 3

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

USENIX

Upcoming Events

2000 USENIX Annual Technical Conference

JUNE 18-23, 2000
San Diego Marriott Hotel & Marina, San Diego, California, USA
Web site: http://www.usenix.org/events/usenix2000

3rd Large Installation System Administration of Windows NT/2000 Conference (LISA-NT 2000)

Sponsored by USENIX and SAGE
JULY 30 - AUGUST 2, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
Web site: http://www.usenix.org/events/lisa-nt2000

4th USENIX Windows Systems Symposium

AUGUST 3-4, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
Web site: http://www.usenix.org/events/usenix-win2000

9th USENIX Security Symposium

AUGUST 14-17, 2000
Denver Marriott City Center, Denver, Colorado, USA
Web site: http://www.usenix.org/events/sec2000

4th Annual Linux Showcase and Conference, Atlanta

Co-sponsored by USENIX and Atlanta Linux Showcase, in cooperation with Linux International
OCTOBER 10-14, 2000
Atlanta, Georgia, USA
Web site: http://www.linuxshowcase.org

First Workshop on Industrial Experiences with System Software (WIESS 2000)

Co-sponsored by IEEE TCOS and ACM SIGOPS (pending)
OCTOBER 22, 2000
Paradise Point Resort, San Diego, California, USA
**Web site:
http://www.usenix.org/events/osdi2000/wiess2000**

4th Symposium on Operating Systems Design & Implementation (OSDI 2000)

Co-sponsored by IEEE TCOS and ACM SIGOPS
OCTOBER 23-25, 2000
Paradise Point Resort, San Diego, California, USA
Web site: http://www.usenix.org/events/osdi2000

14th Systems Administration Conference (LISA 2000)

Sponsored by USENIX and SAGE
DECEMBER 3-8, 2000
New Orleans Marriott Hotel, New Orleans, Louisiana, USA
Web site: http://www.usenix.org/events/lisa2000

6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '01)

JANUARY 29 - FEBRUARY 2, 2001
San Antonio, Texas, USA
Web site: http://www.usenix.org/events/coots01
Submissions due: July 27, 2000

3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)

MARCH 26-28, 2001
Cathedral Hill Hotel, San Francisco, California, USA
Web site: http://www.usenix.org/events/usits01
Submissions due: September 18, 2000

Java™ Virtual Machine Research and Technology Symposium

APRIL 23-24, 2001
Monterey, California, USA
Web site: http://www.usenix.org/events/jvm01
Submissions due: November 1, 2000

For a complete list of future USENIX events, access **http://www.usenix.org/events**

contents

;login: Vol. 25 # 3, June 2000

;login: is the official magazine of the USENIX Association and SAGE.

;login: (ISSN 1044-6397) is published eight times a year by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$40 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Cover photo: USENIX 2025 Technical Conference Keynote Speakers (see page 65)

motd

On Balance

by Rob Kolstad

Dr. Rob Kolstad has long served as editor of *;login:*. He is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

It's graduation season: proms, choosing colleges and jobs, and commencements.

I delivered a commencement speech five or six years ago to a rural high school in Colorado. I fear it wasn't well received. For one thing, I was scheduled after an emotionally wrenching candlelight ceremony in which the seniors presented each of their parents with a rose that symbolized their thanks for their upbringing, and for another, I didn't mouth the standard platitudes: "The world is your oyster." I couldn't even give them unconditional hope for the future; I'm too much of a student of statistics.

I understand that one of the successful West Coast techies gave a commencement speech that same year at some university. He told the students to ignore their parents' advice about hurrying into job, career, family, and life. He is said to have suggested taking that year off to tour Europe or otherwise delve into your psyche. Once you're immersed in what we call "adult life," it is difficult to get off the merry-go-round even for a few weeks, much less for a month or a year. Right after graduation is, arguably, an excellent time to try such a quest.

The Sunday supplement to our newspaper (*Parade* magazine) sported a brief article this last weekend. Over one-third of the teenagers interviewed professed that they would be earning a million dollars by age 40. What career would yield that million? "Professional athlete, doctor, lawyer, computer programmer, independent businessman." The average salary for those professions, all in all, is around U.S. $75,000/year.

If I could give that commencement speech again, I'd try to preach (I think that's what you do at commencements) about "balance."

Balance is so important, and it seems that so many adults (others, too) are bad at it! Almost everywhere I look, I see people with all sorts of monomaniacal preoccupations. For some, it's programming and computing. They program, play games, surf the Web, chat, and so on, to the exclusion of other parts of their life. Sometimes this single-minded energy is devoted to the job: stories of start-ups where people work dozens of hours more than the standard 40/week are legion. For others, it's spirituality, a hobby, being a fan, etc. etc.

Is this a good thing? I think it probably is for a short burst. I don't think it's good to sustain it to the exclusion of social interaction, cultural dabbling, and a large set of other activities.

It seems that balance must be hard to achieve. After all, parents have to devote untold hours to rearing their children. "Internet time" places great demands on technical workers. Interpersonal relationships can eat up inordinate resources to maintain, especially when any other part of life is not going well.

I've been off-balance at many (most? all?) times in my life. Recently, I was devoting what I perceive to be too much time to work. I'm trying to redress the balance now by spending very little time at work.

My stress level is way down, I think. I am, however, still spending all of my time every day (as if something else might happen). The USA Computing Olympiad is currently more than a half-time job. Spring has sprung in the Rockies, and the lawn/garden need attention. I even wasted a whole day last Sunday just to see what it was like. It's weird.

I'm going to try to balance my life. I'll let you know when I achieve that sort of Karma so many have celebrated in words and song.

apropos

Group Therapy

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a consultant on Internet firewalls and network connections and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

I recently attended a meeting of BayLISA, the Silicon Valley group of system administrators that convenes monthly to enjoy a guest speaker and interact with peers. Since I don't make it to the meetings regularly anymore, the spectacle of the now-traditional opening announcements moved me much as the first meetings in the spring of 1991 had.

I'd guess that I first heard about the first BayLISA meetings through USENET news, but I can't remember for certain. I do remember that I circulated the announcement among my co-workers, and one of them humored me and took the 40-minute ride down to the evening meeting in Silicon Valley with me.

I had very specific reasons for attending that evening. At the time, I was managing the UNIX systems-administration team at Lawrence Livermore National Laboratory, and I was in an ongoing battle with the computer science department over hiring system administrators. They wanted to hire system administrators as computer associates (usually nondegreed, hourly employees), and I wanted to hire them as computer scientists (the Lab title for degreed, salaried professionals). I was continually conducting my own "salary surveys" and collecting job descriptions to back my position. I was vocal and unpopular with management for my stance on the issue, but I maintained that system administrators were professionals, and that the LLNL job classifications were archaic and broken. I felt my team members' value wasn't being recognized, and I wanted to talk about it to anyone who would listen to me.

I remember images from my first meeting vividly. The attendance wasn't huge. The group was still forming, and a good portion of the meeting was discussion among the attendees about organization, purpose, and common concerns. Most important for me, they touched my "hot button" of misclassification of system administrators in the workplace. Many of them were experiencing the same dilemma I was, and it was good therapy for me to hear I wasn't alone in my struggle. I tried to contain my excitement during the meeting, but I was perched on my soapbox preaching to my fellow admin the entire drive home. I was empowered by the access to my peers, and I was hooked.

The meetings grew in popularity. They became more structured as invited speakers were introduced. Still, the opening announcements at each meeting provided fodder for the correct-classification cause as people began to announce they had job openings for qualified system administrators, and it became clear that demand was exceeding supply. To me that represented much-needed leverage for the uphill battle against job misclassification; it's easier to negotiate when you're in the driver's seat! Later the formation of SAGE provided the necessary credibility for system-administration job descriptions so that they could be widely accepted and end the debate. The BayLISA meetings and opening announcements continue to act as entertainment and group therapy combined. So many jobs openings are announced that each one is greeted with affirming giggles from the audience. Last month's announcement from an individual seeking a job immediately launched a tongue-in-cheek hiring war from the audience, with offers of laptops, stock, and flexible work environments to seal the deal. Each additional offer brought a roar of approval from the crowd.

To the collective voice of the systems-administration community's credit, BayLISA opening announcements describe a position simply as a "SAGE Level III," and everyone knows what that means. It's clear that the power in numbers of like-minded professionals has been effective in furthering the causes of system administrators. It's provided the catalyst for change and the power to back it up. This year SAGE is updating the *SAGE Job Descriptions* booklet to reflect today's modern computing environment. If you have suggestions for the update, please send them to *<jobs-update@usenix.org>*.

This issue's report is on the 7th USENIX Tcl/Tk Conference, held in Austin, Texas, February 14–18, 2000.

OUR THANKS TO CLIF FLYNT FOR THE SUMMARIES AND THE PHOTOGRAPHS.

Conference Reports

7th USENIX Tcl/Tk Conference
February 14–18, 2000
Austin, Texas, USA

Summarized by Clif Flynt

If the 6th Tcl/Tk Conference was Tcl's "coming of age" conference, then the 7th was "more of the same, but bigger." For example, at the last conference people described systems used to load-test and validate pilot-scale networks; this year, we learned about the design for a system that load-tests and validates equipment on the largest telephone network in Europe.

The papers presented were excellent, though the range was not as large as in previous years. Also, the attendee list seemed more skewed toward industry than academia. This may be another sign of Tcl's increasing acceptance in industry, or it may just reflect that moving the conference to the spring (midterms, instead of the previous summer-vacation date) made getting time to attend, as well as completing projects and writing papers before the submission deadlines, difficult.

Along with the technical papers, this meeting featured a "Texas Shootout" Tcl coding contest, with some amazing coding examples; an after-dinner Tcl trivia game show (So, You Want to be a Tcl-Hacker); receptions sponsored by Scriptics, Vignette, and (I think) others; and of course, evening BoFs.

The Austin Marriott was an excellent hotel for this event. The function space was large and centrally located. The hotel snack bars and restaurants were convenient, and the lobby area had plenty of sitting and talking spaces. Folks with more time could take a short walk to view the Texas Capitol building or find some truly excellent restaurants.

As usual, the best parts of the conference for me were getting face time with folks that I've just exchanged mail with in the past, finding out what people are doing, and making plans for what we'll do next.

KEYNOTE ADDRESS

TCL IN AOL DIGITAL CITY: THE ARCHITECTURE OF A MULTITHREADED HIGH-PERFORMANCE WEB SITE

Jim Davidson, America Online, Inc.

Jim Davidson described the AOL Digital city, a mammoth database-driven Web site, and the AOLServer HTTP daemon that it extends.

Jim Davidson

AOLServer, derived from the NaviServer HTTP server, was purchased by AOL when the company moved from being a dial-up-based company to being an Internet-based company. The AOLServer server is now an open-source stem, available at *<http://www.aolserver.com>*.

One of Davidson's first actions when he joined AOL was to integrate Tcl into the NaviServer/AOLServer. This provides a factor-of-100 speed-up over traditional CGI.

AOLServer supports both embedding code in HTML pages and embedding HTML generation in code modules.

The AOLServer is a multithreaded process, and Davidson discussed some of the pitfalls of multithreaded processes, including mutexes, locks, shared memory handling, resource allocation, and processing events. In early versions of the AOLServer, these problems were solved with patches and extensions to the Tcl code. AOL modified the Tcl slave interpreters to make copies of some data structures instead of having them shared. This created a working system but made migrating to more modern interpreters difficult.

Davidson described the architecture for AOLServer 3.0, which uses a cloning model to create new interpreters and use more of the features built into Tcl. Whereas migrating from Tcl 7.3 to 7.4 took a month of effort with AOLServer 2.1, migrating from 8.2.2 to 8.2.3 took 15 minutes with AOLServer 3.0.

AOLServer 3.0 implements locks and mutexes at the script level instead of the interpreter level. When AOLServer acquires pages, it caches them internally. This makes AOLServer resistant to the "Slashdot effect": If a page is being hit often, it remains in cache, instead of needing to be regenerated.

Despite all the nice features of AOLServer, it's still just an HTTP server. The useful and interesting content is created with the AOL Digital City modules. Digital City is a set of Tcl modules that hit multiple databases to generate custom dynamic HTML pages on the fly. One outcome of this is that pages are viewed as collections of text, instead of as a single entity.

Generating pages like this can be expensive because of the number of separate databases and tables that must be queried to build a page. Because of this overhead, much of the effort in the development of this package has been spent tuning the caching mechanisms. One problem is that as the number of combinations of dynamic information increased, it turned out that caching pages didn't work. Each page was likely to be unique, and so needed to be created from scratch.

The current solution is to put a high-speed front-end KSAM database between the Digital City modules and the complete (slow) relational DBMS. The pages are still built dynamically, but the most-used pieces of information are kept in the high-speed front-end database.

Davidson described some of AOLServer's many logging features. For example, the AOLServer maintains a count of how often commands are used. For example, the most commonly used commands in Digital City are if, set, and string. (This is not surprising in a site that is producing heavily customized text pages.)

One of the unexpected advantages Tcl brought to the Digital City project is that the simplicity of Tcl makes it easy to bring people up to speed quickly. More information on AOLServer is available at <*http://www.aolserver.com*> and <*http://www.photo.net*>.

REFEREED PAPERS

SESSION: MIDDLEWARE

The Middleware session papers demonstrated several different patterns for using Tcl to glue programs together. These patterns ranged from using Tcl with CORBA, to agent-style distributed applications, to techniques for converting standalone Tcl applications into collaborative-style distributed applications.

RAPID CORBA SERVER DEVELOPMENT IN TCL: A CASE STUDY

Jason Brazile and Andrej Vckovski, Netcetera AG

Brazile described a middleware data-distribution project that uses a Tcl kernel to access multiple sources and combine and distribute that data packaged in different formats. This package acquires data in various proprietary binary formats and converts it to Tcl data structures for internal use. It saves processed data in an internal cache and distributes the data formatted as XML documents, as CORBA objects, or even as Tcl data structures.

Jason Brazile

The architecture they decided on for this project – mapping CORBA methods to Tcl procedures – provided the extra benefit of allowing the core functionality of the server to be developed and tested in Tcl without needing a CORBA development environment or network infrastructure.

Another fallout from this architecture is that CORBA stubs, skeletons, and Tcl/C++ conversion routines could be generated automatically from locally developed Tcl code-generation scripts according to a given IDL specification.

Advantages of using Tcl and scripting included:

- Robustness
- Ability to develop and test outside of CORBA infrastructure
- Easy reconfiguration when data formats changed
- Easy rebuilding when the CORBA IDL was changed (12 times in 12 months)
- Ease of testing
- Speed of development (5 months from plan to deploy)

AGNI: A MULTI-THREADED MIDDLEWARE FOR DISTRIBUTED SCRIPTING

M. Ranganathan, Mark Bednarek, Fernand Pors, and Doug Montgomery, National Institute of Standards and Technology

M. Ranganathan described AGNI (AGents at NIst, and Sanskrit for fire), a package for scripting reconfigurable event-oriented distributed systems. This package allows clients and servers to migrate from site to site to balance load and access particular resources.

AGNI has the concept of mobile streams, which Ranganathan described as being like active mailboxes. This provides a single point of control that knows how to relay messages to the appropriate clients and servers.

To support the distributed nature of AGNI, Ranganathan and his associates built a reliable protocol on top of UDP. This was necessary to allow applications to migrate from machine to machine, since TCP doesn't like to have endpoints change during a session.

Ranganathan demonstrated how standalone applications can be merged into AGNI, and compared AGNI to other distributed systems such as Tcl-DB, AgentTCL, and SUMATRA. In brief, AGNI is more versatile and supports more models of distributed processing.

Introducing QoS Awareness in Tcl Programming: QTcl

Roberto Canonico, Maurizio D'Arienzo, Simon Pietro Romano, and Giorgio Ventre, Università di Napoli "Federico II"

Part of the future of the Internet is that applications will be able to reserve a certain bandwidth (Quality of Service) between client and server.

Canonico described a Tcl extension that allows scripts to use the RSVP protocol to request bandwidth in a manner that is compliant with the SCRAPI programming interface. This new facility is built on top of the Tcl-DP package. Having this package available allowed them to concentrate on the SCRAPI portion of their project, rather than also having to develop a distributed processing vehicle.

One of the requirements for a QoS-aware application is that it must be able to estimate its bandwidth requirements.

Canonico described the QTcl API, explained the rationale for using Tcl-DP, and showed experimental results that demonstrate how an application that has reserved a particular bandwidth can continue to function properly, even in the presence of other net traffic causing congestion in the network's routers. Of course, in order to obtain this behavior, special routers with QoS support and RSVP enabled need to be deployed within the network.

This package has been implemented and validated on a test network equipped with FreeBSD boxes acting as routers. The QoS support in these boxes was achieved by implementing a WFQ packet scheduler as an extension of the ALTQ package.

The QTcl package (for UNIX-based systems) is available at *<http://www.grid.unina.it/qtcl>*. The authors are currently implementing the QTcl interface for Windows-based systems.

CollabWiseTk: A Toolkit for Rendering Stand-alone Applications Collaborative

Hemang Lavana and Franc Brglez, North Carolina State University

Lavana proposed an architecture that allows standalone Tk applications to be transparently converted to distributed, collaborative applications. The keystone of this architecture is techniques that allow the distributed widgets to mediate synchronization and data.

Two users can interact with text widgets in several ways:

- Both users can have view-only access to the text, with individual cursors.
- The text displays can be linked so that each user is viewing the same section of text.
- One user can be locked out while the other has write access to the text.
- Each can have a separate cursor and write access to the text. Each display will show where the user is currently working.
- There may be two text widgets, each associated with a user. Users may type in one and view what the other user is typing.

Lavana explained how the appropriate interaction model varies depending on the application. For example, group conferencing, shared whiteboard, shared editing, and chat each requires a different interaction model.

The CollabWise architecture is to have a central group server attached to multiple application servers. Each application server may have multiple clients. The group server distributes the interaction models to the clients.

The interaction models are implemented by replacing the usual Tk widget bindings with an appropriate set of distributed bindings. This transparently converts an application written for a single platform into a distributed application.

More details about this project are available at *<http://www.cbl.ncsu.edu/software>*.

SESSION: TESTING AND INTEGRATION

The three papers in this session demonstrated the wide range of applications for Tcl/Tk. Seeing what can be done when you embed a full scripting language into a debugger makes you wonder why all debuggers haven't been built like this for years, while the papers on load-testing a telephone network and gluing together legacy systems show how well Tcl/Tk can scale to world-scale problems.

GDBTk: Integrating Tcl/Tk into a Recalcitrant Command-line Application

Jim Ingham, Cygnus Solutions

Jim Ingham described ways that an embedded interpreter can make a debugger more useful. The debugger he used as the example is GDBTk (now known as Insight), being used to debug a Tcl interpreter running a Tcl script.

Traversing a stack and listing the call parameters, including pointers, is an annoying and time-consuming part of debugging from a core dump. In particular, when debugging an interpreter and trying to determine what part of a script caused an error, you need not only to examine the C stack, but also to find the internal interpreter execution stack.

Traversing the call stack, dereferencing pointers, and displaying the interpreter stack is an application that can be easily automated. In fact, any set of repeated operations is a candidate for becoming a Tcl script.

Some advantages that Tcl has over the GDB macro language are the looping and test commands. These allow a programmer to automate such tasks as stepping through a linked list or array looking for a particular data pattern.

GDB events (breakpoints, touching variables, etc.) can be treated as Tcl events, invoking a Tcl proc to examine the program state and decide what to do from there.

The GDBTk/Insight package is available at *<http://sourceware.cygnus.com/insight>*.

Tcl/Tk: A Strong Basis for Complex Load Testing Systems

Ahmet Can Keskin, Till Immanuel Patzschke, and Ernst von Voight, Patzschke + Rasp Software AG

One of the environments in which Tcl/Tk has proven very useful is automated testing. Till Patzschke described using Tcl/Tk to performing load testing for Deutsche Telekom. The package they developed measures all the components of the network, from the user's connection, to the telephone network (via modem, ISDN, DSL, etc.), to the remote server. This system duplicates all the actions a user would perform: dialing out to the network, browsing, and so on.

Patzschke pointed out that one advantage to building a set of automated scripts over hiring a number of students to follow a written script is that the automated scripts provide much more reproducible results.

Another advantage is that a subset of the scripts can be run continuously to provide early warning if the network performance should begin to degrade.

Till Patzschke

The initial driving requirement for this system was to validate hardware and software installations. When acquiring new hardware, the contracts specified that problems found within 30 days would be fixed under the initial contract, but problems found later would be fixed for a fee.

This was a strong impetus to developing a reliable automated (fast) load-testing system.

After evaluating several OS and language choices, Patzschke and his associates settled on Tcl/Tk and Linux. He noted that these choices allow the developers to quickly adapt testing scripts as requirements change and to scale the system as necessary by adding more commodity hardware.

The system is operated from a single Tk application that controls the test stations. More than 60 systems are being controlled by the single Tk front end.

More information on the Automated-User package may be obtained from *<http://www.internetwork-ag.de>*.

Using Tcl to Build a Buzzword-Compliant Environment that Glues Together Legacy Analysis Programs

Carsten H. Lawrenz and Rajkumar C. Madhuram, Siemens Westinghouse Power Corp.

Like many companies, over the years Siemens Westinghouse has accumulated a number of software tools for developing engineering designs. However, the paradigm for how tools should be built, both user-interface and data formats, changes over time, and these changes make it difficult to use tools from different generations together.

To continue being usable, the tools must either be rewritten (opening the gate for introducing new bugs), or the old applications must be glued together with wrappers to present a uniform interface to the users and translate between different data formats.

Rajkumar Madhuram described a project that created a Tcl-based set of wrappers to integrate the tools Siemens uses to design electrical generators into a unified tool that presents a single interface to the user.

The goals for this project included reducing the number of times data had to be translated from one format to another by hand (and the errors that can introduce) and reducing the time it takes to design a generator. Tcl was chosen for this application because of its runtime speed, ease of development, and rich set of GUI widgets.

Since some tools run on specific hardware, the Siemens Integrated Design (SID) package uses a client-server architecture, in which a thin client runs on the designer's local workstation, while the data and analysis tools exist on remote servers. The SID package generates user-input screens dynamically, using the Tk entry widget for keyboard input and allowing users to interact with canvas graphs to view and modify other data.

SESSION: WEB TECHNOLOGIES

The early driving force for the Web was gluing together all the disparate data sources such as FTP, WAIS, Gopher, and documents. One of Tcl's strengths is gluing together data sources such as programs, files, and databases. With this philosophical connection to start from, it's no surprise that there are a lot of Tcl Web-based applications, ranging from HTTP servers to browsers embedded in Tk widgets.

Proxy Tk: A Java Applet User Interface Toolkit for Tcl

Mark Roseman, TeamWave Software, Ltd.

Many computer users are most comfortable with browser-based user interfaces. While JavaScript and HTML are adequate for forms-based applications, they are inadequate for complex program interactions.

Larger applications can be written as applets running inside the browser. This allows a program to present a familiar interface to the user while providing the functionality developers require. Writing a large, complex application as a Java applet has other problems, including large download times and the slow development pace for low-level systems languages like Java and C++.

Once a developer has gotten used to the speed with which applications can be developed with a high-level language like Perl or Tcl/Tk, they are unwilling to move back to low-level languages like Java and C++ for the main program flow. Using the Tcl plug-in as the applet vehicle solves these problems, but many users and system administrators are resistant to adding new plug-ins to their environments.

Roseman described a solution to this dilemma: a thin Java client to generate GUIs that runs within a browser without requiring a plug-in download. The main program logic resides on the remote server, and only the GUI client runs within the browser.

The thin client includes a simple parser, wrappers around the AWT widgets to generate a GUI, and functionality for sending events back to the server. At the server end, traditional widget-interaction commands are replaced with procedures that support remote operation.

Roseman's group has found this to be a good compromise, allowing them to do their primary development and debugging in the traditional Tcl/Tk environment and to distribute the product with a small applet.

Roseman noted that not all Java runtime engines are created equal, and that Java applets can be described as "Write Once, Debug Everywhere." Having only the thin client in Java made the multiple-platform deployment much easier, since there was relatively little Java code in the product.

The TclHttpd Web Server

Brent Welch, Scriptics Corp.

Brent Welch described an HTTP server written entirely in Tcl. This server has been in use at Sun Microsystems for several years and is also the server for *<http://www.scriptics.com>*.

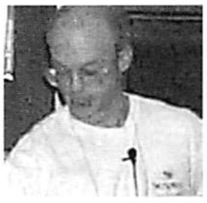
One of the unique features of this server is that it can be embedded into other Tcl applications, such as the TclPro license server, the Lyris MailShield, and NationsBank network-management applications.

The TclHttpd server supports dynamic content by:

- embedding Tcl scripts into HTML pages
- allowing a template file to be associated with each HTML file
- mapping URLs to Tcl procedures

Other internal functions include support for checking form data, and automatic mapping of attribute=value pairs into procedure arguments (using Tcl's introspective info command to determine what a procedure's arguments are).

Using Tcl scripts instead of CGI to handle form data removes the need for forks, reduces the load on the hardware, and improves the performance of the TclHttpd server.


Brent Welch

Welch showed some timing studies which demonstrated that the TclHttpd has decent performance. While the TclHttpd server is slower than the best server in each test, it is faster than the worst. The server can handle about 100 hits per second and has proved extremely stable.

This package is available at *<http://dev.scriptics.com/software/tclhttpd/>*.

TkGecko: A Frill-Necked Lizard

Steve Ball, Zveno Pty Ltd.

Steve Ball described his effort to embed the Mozilla NewLayout HTML/XML viewer inside a Tk widget. This effort is hampered by many features of the Mozilla code, including Mozilla's use of Gtk (which has its own event loop), the sheer size of the Mozilla package (compiling Mozilla requires 1 GB of disk space), the speed with which the Mozilla code base evolves, and the lack of current documentation.

Fortunately, the entire Mozilla package is not required for this effort. The NewLayout viewer module is a small subset of the browser code. Ball described the architecture of the NewLayout module, how it interacts with the Mozilla browser, and how these interfaces can be used to interact with the Tcl interpreter through the callbacks.

Ball demonstrated that the widget works. Future effort will be directed toward further integration of the NewLayout callbacks and the Tcl interpreter.

Scriptics Connect: An XML Integration Server Based on Tcl

Eric Melski, Scott Stanton, and John Ousterhout, Scriptics Corp.

Stanton described the application developed by Scriptics to facilitate creating XML-based business-to-business applications.

XML provides an easily extended, easily interpreted basis for information exchange. Having an easily parsed data-description language makes it easy for an expert programmer to generate a parser and create an application. It would be better if applications could be created by less skilled users.

Scott Stanton

The Scriptics Connect package allows a relatively untrained user to describe how the fields on a form should be treated with a Post-it Notes paradigm: fields in the displayed form can be marked and actions associated with the data in that field.

The Scriptics Author program displays a tree representation of the document skeleton. A user can attach scripts to the document elements without needing to navigate the document. After the user has selected form elements and described how values associated with those elements should be processed, a parser for this form is generated automatically.

Scriptics provides prebuilt data handlers for converting an XML form into the format appropriate for internal use.

More details about Scriptics Connect are available at *<http://www.scriptics.com>*.

SESSION: USER INTERFACE AND APPLICATIONS

Because the basic Tk GUI toolkit is rich and easy to extend, Tk has become one of the languages of choice for experimenting with user-interface designs. These papers introduce a novel way of displaying data with animations, experiments in collaborative computing, and a different way to view packaging scripts.

Supporting Information Awareness Using Animated Widgets

D. Scott McCrickard and Q. Alex Zhao, Georgia Institute of Technology

As has been noted, modern man is constantly bombarded by information. Scott McCrickard and his associates are investigating techniques for making information available on a CRT without distracting the user.

One of the techniques they are exploring is forms of animation: fading one datum into another and scrolling information horizontally (like a ticker tape) or vertically (like lines on a teletype). The goal is to modify the displayed information without jarring the user, while still notifying the user that new information is available.

As a research tool, they created several new Tk widgets that support images fading from one to the next, as well as scrolling text. These new widgets follow the standard Tk widget conventions, such as associating variables with widgets. For example, a -textvariable is associated with the ticker widgets, just as a -textvariable can be associated with a label. When new data is saved in the variable, the display automatically reflects the new values.

These animated widgets can be synchronized with one another. This lets a designer use a set of logos and scrolling text, with the logo providing context information while the scrolling data provides details.

D. Scott McCrickard

One problem with changing displays is that the important information may be, not the actual data, but the change from a previous value (for example, stock prices). When the user is alerted that data has changed, she may not recall the previous data. In order to retain history, old information can be displayed in a widget as shadows or background text. This lets a user see that something has changed and what the old value was.

McCrickard demonstrated several of these widgets, showing how the package can be used with applications ranging from a sports-score display to a grad-students employment listing.

The automated widget set is part of a package for designing agents that is available at *<http://www.cc.gatech.edu/~mccricks/agentk>*.

Collaborative Client-Server Architectures in Tcl/Tk: A Class Project Experiment and Experience

Franc Brglez, Hemang Lavana, Zhi Fu, Debabrata Ghosh, Lorie I. Moffitt, Steve Nelson, J. Marshall Smith, and Jun Zhou, North Carolina State University

It's generally recognized that large projects should be used as part of the computer science curriculum. Franc Brglez described a large class project in which students worked together to develop an application that would allow them to jointly write a paper.

Franc Brglez

Brglez described the traditional approach to a number of people writing a paper – each student would have a login account on a central machine or would email their writings to the person collating the works. For this application, it was decided to use a client-server Web-based approach.

Tcl/Tk was chosen as the vehicle for this project because the power and simplicity of the Tcl channel architecture makes it easy to write client-server applications and because the language is easy to learn (an important requirement for a single-semester project).

The server for this application supports user validation, file ownership and protection, execution of system commands on the server, and file upload/download.

A student with prior Tcl experience wrote a universal server, and the students each created a GUI client to interact with the server. To validate the concepts, students used their clients to write and revise individual contributions to the paper.

The client-server architecture initiated in this class project continues to evolve a number of user-configurable collaborative environments.

More information about these collaborative computing projects is available at:

> *<http://www.cbl.ncsu.edu/publications/#2000-TclTk-Lavana>*
> *<http://www.cbl.ncsu.edu/publications/#2000-TR@CBL-02-Brglez>*
> *<http://www.cbl.ncsu.edu/publications/#2000-TR@CBL-03-Lavana>*
> *<http://www.cbl.ncsu.edu/software/>*

Scripted Documents

Jean-Claude Wippler, Equi4 Software

Writing multiplatform applications in Tcl/Tk is easy. Distribution is not always so simple.

Jean-Claude Wippler described the Metakit Tcl extension, which merges a small database into the Tcl interpreter and keeps application scripts and associated data within the database.

The advantages of this technique include:

- Data and code that understands the data cannot become separated.
- A single binary can be used for multiple applications.
- Access to the code/data can be controlled.
- All code and data modules are wrapped in a single runtime file.
- Since the data and code are combined, there is no need to track installation and data paths.

Metakit has been in production use for a couple of years. One example of a Metakit-packed application is the *Tcl'ers Wiki* at *<http://purl.org/thecliff/tcl/wiki/>*.

More information on Metakit is available at *<http://www.equi4.com/metakit/>*.

SESSION: EXTENDING CORE TCL

The easy-to-extend and easy-to-read nature of Tcl makes the Tcl core a popular target for optimizations and experimentation. This constant activity has a lot to do with Tcl's application to a broad variety of problems.

Papers in this session discussed efforts at unifying some of the development, ways to make Tcl more object-oriented without copying C++/Java, and ways to make the Tcl core even more versatile.

The Tcl Extension Architecture

Brent Welch and Michael Thomas, Scriptics Corp.

Brent Welch described the problems involved with distributing the packages and extensions for Tcl.

Tcl scripts are easy to use in a multi-platform environment, but compiling Tcl extensions is not always so easy. The individual authors use their own conventions, and different platforms support different tools (VC++, AutoConf, Imake, etc.).

The TEA initiative is dedicated to developing a set of techniques and templates for developers to use when designing an extension and for creating configuration and installation scripts. The goal is to make Tcl extensions more consistent in form and easier to develop.

The current version of the TEA framework includes a set of M4 macros to simplify building an autoconf file that can be used on UNIX, or on Windows with the CygWin Tools.

Welch also discussed the stubs library, which was introduced in Tcl 8.1. The stubs library makes it possible for Tcl interpreters to load shared libraries at runtime, whether or not the underlying OS provides support for dynamic libraries. This is done via a stubs table, similar to the reference tables used to load .so files under Solaris or Linux.

Two byproducts of the stubs tables are that an extension can be compiled once and used with interpreters at different revision levels, and that the stubs tables define a "blessed" API into the Tcl interpreter that won't change.

As part of the TEA testing, Scriptics is now doing nightly automated builds of several extensions and packages, including BLT, Expect, Itcl, TclX, OraTcl, and TclODBC.

Some of the changes that fall out from using the new TEA configuration architecture (such as possibly removing TclConfig.sh) generated spirited discussion from the audience.

The TEA homepage is *<http://www.scriptics.com/tea/>*. A sample extension is available at *<ftp://ftp.scriptics.com/pub/tcl/examples/tea>*.

XOTcl – An Object-Oriented Scripting Language

Gustaf Neumann, Vienna University of Economics and BA, and Uwe Zdun, University of Essen

Gustaf Neumann

Gustaf Neumann described XOTcl (pronounced exotickle), an object-oriented programming extension to Tcl that enhances Tcl's properties rather than imposing the C++/Java worldview on Tcl.

The Extended Object Tcl classes are designed to support introspection and extensibility, as well as protection and data-type checking.

The introspective nature of Tcl is one of the features that distinguishes it from other languages. For example, a running Tcl program can get a list of all the defined procedures and their bodies and arguments. On the other hand, most OO languages believe that the internal nature of an object should be hidden from all other objects. With XOTcl, the objects can be queried regarding their internal state, just as other Tcl and Tk objects can be queried.

Another way that Tcl differs from most OO languages is the ease with which program behavior can be modified at runtime. The behavior of most OO class methods is defined at compile time. Tcl takes advantage of its interpreted nature to allow new procedures to be added, or the class structure changed, as conditions dictate.

XOTcl objects can be extended at runtime via filters (at the class level) and mixing classes (at the object level). These methods provide sets of new procedures and interceptors for existing procedures for an object for behavior customization. These interceptors can be used for a wide range of situations, including debuggers and high-level OO constructs such as design patterns.

While adding features to support Tcl's strength, Extended Tcl also includes the common object-oriented language features of inheritance, abstract classes, and assertions.

XOTcl was developed from the MIT OTcl package and is available from *<http://nestroy.wi-inf.uni-essen.de/xotcl>*.

A Multi-Threaded Server for Shared Hash Table Access

Andrej Vckovski and Jason Brazile, Netcetera AG

Vckovski and Brazile are two of the developers of a multiple-feed, caching front end for a data server. This system required a high-speed but not complex database solution. The data needs to be cached until it is overwritten by fresh data and to be served to clients on demand.

The speed requirements made a traditional disk-based DBMS solution unacceptable. A requirement that the application be adaptable to changing data formats made an in-memory database supported by a compiled language like C or Java too inflexible. These two requirements led Vckovski and Brazile toward a scripted solution.

Vckovski described a solution in which multiple computers accept the data feeds and cache the data using the Tcl hash tables. These hash tables are shared across the multiple threads and systems so that clients can access the data transparently.

The system has been in production with data shared across four systems since summer 1999. In this configuration it handles several hundred updates per second.

Vckovski summarized that the Tcl thread safety is real and robust. He also observed that the Tcl hash algorithm is well optimized, and even with large tables (about 250,000 entries) the search distances are small.

Feather: Teaching Tcl Objects to Fly

Paul Duffin, IBM Hursley Laboratories, UK

One of Tcl's failings is the lack of complex data structures. Paul Duffin believes he has a solution to that problem (and perhaps all your problems, depending on what your problems are).

Tcl's internal data representation was changed in revision 8.0 from simple NULL-terminated strings to an object with reference counts, and both textual and native form representation.

Paul Duffin

Duffin described an extension to the Tcl object that allows multiple interfaces to objects. With multiple views, it becomes possible to use objects in more ways than traditional Tcl supports.

Duffin described a large number of data constructs that he has built using the Feather extension, including generic collections, complex lists, maps, structures, opaque objects, and even the functional programming concepts of lambda and curried objects.

Duffin points out that the range of new data structures he created shows the versatility of this approach, making this extension a good option for others who need data structures that Tcl does not natively support.

SESSION: WORKS-IN-PROGRESS

D. Richard Hipp described his HTML display widget. This widget will display all HTML tags (including ugly nested tables) but does not support style sheets.

Steve Landis revisited the Software Tools paradigm and demonstrated a cross-platform, GUI-driven technique for linking programs with virtual pipes.

Glenn Thomas described a project using Tcl to automate a set of legacy programs. This packages executes the applications from a browser using the Tcl plug-in and Expect, and converts report output to HTML for easy viewing.

Dave Beasley described the latest changes to SWIG, a package that will convert a set of header files into a Tcl extension. The new changes create much smaller extensions.

Roy Terry showed a Tk application called *hits* that displays a file and searches for words that match a pattern. The application highlights the patterns and also builds a map showing where the hits are within the file.

Gil Benson described a user-friendly QDDB-based testing system that links to his organization's trouble-ticket system.

Leo Schubert described a set of wrappers for the native Tk widgets for Windows, Gtk, and KDE. Among other features, these wrappers allow new attributes to be added to widgets.

Devin Eyre uses Tcl/Tk and GIS information to create weather maps.

David Vice described how CPU uses Tcl as a data bus to merge multiple data sources and applications. They use XML as a standard for moving data between the objects.

Jeff Davis described his Tcl-based Web interface to a database for a fantasy-role-player baseball league.

Clif Flynt's laptop described a new method for creating multimedia applications in Tcl, while Flynt demonstrated that humans are not required for a presentation.

David Russell described several tools that he uses with Scotty to simplify systems administration.

Brad of Neosoft described a Tcl interface to LDAP. It's available at *<http://www.openldap.org>*.

Bill Krimmell described a Tcl interface to some National Scientific interface cards running under Linux.

Steve Ball showed TclXml working with the Xerxes C XML library, a set of code he debugged on the plane coming to the conference. This demonstrates that flying halfway around the world does have some benefits.

Mark Harris explained how Tcl is being used to generate dynamically constructed HTML pages in China.

Brian Griffin of Model Technology described a VHDL/Verilog simulator with a user interface written in Tcl driving a C code engine.

using java

Active Content in Java

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the devel-opment of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

<prithvi+@ux4.sp.cs.cmu.edu>

Drawing and animation are integral parts of writing Java applications. Applets frequently contain various AWT (Abstract Windowing Toolkit) Components that are part of a graphics context. When they need to be redrawn, the AWT starts with the top Component in the hierarchy and works its way down to the bottom.

The purpose of this article is to expose the reader to various capabilities that are part of the "Active Content" in Java. I'll focus on issues related to the drawing model, drawing shapes, the graphics context, drawing text, measuring text images, and loading them both synchronously and asynchronously. I'll also touch briefly on eliminating flashing and animating images. This information provides for a richer knowledge base from which to venture into writing more interesting Java programs with active content.

The Drawing Model

Redrawing is performed when the AWT starts with the topmost Component in the hierarchy and works its way down to the bottom Component. Given that containers are Components as well, this applies equally to applets and panels.

Each Component draws itself before it draws any of the Components that it contains. This ensures that a Panel's background is visible only where it isn't covered by one of the Components embedded in it.

Programs draw only when the AWT tells them to do so. For instance, this can happen when a Component becomes uncovered by the user. Another example is when there is a change in the data being reflected by the Component.

The AWT requests that a Component draw itself by invoking the Component's update() method. The default Component implementation of update() clears its clipping area in the current background color and then calls the paint() method. The default implementation of paint() does nothing.

Another way to look at this is that a programmer causes a refresh that's due to some change in state of the control by calling repaint(), which in turn calls update(). It is usually the case that the paint() method is overridden to provide component-specific drawing. It is also possible to override update() for more sophisticated results. However, it is possible to call paint() directly without calling update(), so paint() must always be implemented.

Note that update() sets the clipping region, and so if the object being drawn changes between drawings, it is possible to end up with half of one and half of the other.

As can be anticipated, the paint() and update() methods must execute very quickly to ensure acceptable performance. (Note: This is where it becomes advantageous to use threads with applets to achieve better performance. My December 1999 "Using Java" article, "Using Threads Within Applets," discusses this topic.)

The Graphics Context

The argument to the paint() and update() methods is a Graphics object that represents the context in which the Component can perform its drawing. The Graphics class provides for drawing and filling rectangles, arcs, lines, ovals and polygons, text and images.

It is also possible to "set" and "get" the current color, font, or clipping area, and to set the paint mode.

The Graphics class has a multitude of methods that permit the drawing of various shapes.

The Color class encapsulates an "r.g.b" triplet, and color is set by:

```
Graphics.setColor(Color color);
```

How text is drawn is defined by the Font class that encapsulates a font (e.g., Times Roman or Helvetica). It also supports point size and a style, although currently only bold, italic, and plain are supported. An example of setting the font is:

```
Graphics.setFont(Font font);
```

Drawing Shapes and Text

The Graphics class has a multitude of methods that permit the drawing of various shapes, including:

```
Graphics.drawRect(int x, int y, int width, int height);  for drawing rectangles
Graphics.fillRect(int x, int y, int width, int height);   for filling a rectangle
Graphics.drawRoundRect(int x, int y, int width, int height, int arcwidth, int archeight);
Graphics.fillRoundRect(int x, int y, int width, int height, int arcwidth, int archeight);
```

When drawing text it is better to first consider whether it is possible to use a text-oriented Component such as the Label, TextField, or TextArea class. The alternative is to use drawBytes(), drawChars(), or drawString(). For example you can "draw" a string as follows:

```
Graphics.drawString("This is an example of drawing a string", x, y);
```

where x and y specify the coordinates for "drawing" the text.

Measuring Text

In order to determine if text can fit inside a certain area, it is necessary to query the characteristics of the Font using getFontMetrics(), which returns a FontMetrics object corresponding to a given Font:

```
FontMetrics foo = Graphics.getFontMetrics(Font f);
```

Some of the most commonly used methods that are part of the FontMetrics class are charWidth(), getHeight(), getAscent(), getDescent(), and stringWidth().

Images and Loading Images Asynchronously

All images are encapsulated by the image class. There are two ways to load images, Applet.getImage(URL) and Toolkit.getImage(filename/URL). Both of these load an image, and return an Image object. The method getImage() returns immediately without checking whether the image data exists or whether it's been successfully loaded. This is done to improve performance, since it is not necessary to wait for an image to be loaded before going on to perform other functions in the application. Some examples of loading images are:

```
Image image = Applet.getImage(getDocumentBase(), "Image.jpeg");

Image image = Toolkit.getDefaultToolkit().getImage( new/URL
            ("http://www.sample.com/images/sample.gif"));
```

In order to draw the image it is necessary to use one of the Graphics.drawImage() variants. In other words, drawImage() allows you to specify the image to draw and the positioning and scaling of the image. It also allows the specification of the color to draw underneath the image. This is useful if the image contains transparent pixels.

The ImageObserver parameter specifies an object that implements the ImageObserver interface. This object will be notified whenever new information regarding the image becomes available. The Component class implements the ImageObserver interface to invoke the repaint() method as the image data is loaded. The method drawImage() returns immediately even if the image data has not been completely loaded; this results in the image being displayed partially or incrementally.

The easiest way to track the loading of images and to make sure that drawImage() draws only complete images is to use the MediaTracker class. The sequence is:

1. Create a MediaTracker instance.
2. Tell it to track one or more images.
3. Ask the MediaTracker the status of images.

The following code examples elucidates this point:

```
tracker = new MediaTracker(this);
images = new Image(num_images);
for (int i=0; i<num_images; i++) {
images[i] = getImage(this.getDocumentBase()."image" + i);
tracker.addImage(images[i],i);
}
```

and then later in the program

```
for (int i=0; i<num_images; i++) {
     this.showStatus("Loading image: "+i);
     tracker.waitForID(i);
     if(tracker.isErrorID(i)) {
        showStatus("Error loading image"+i);
        return;
     }
}
showStatus("Loading images done.");
```

Animation and Thread Management

Animation is perceived motion accomplished by sequencing rapidly through frames. Frames can be incrementally different images or graphics operations. A good rule of thumb is that animation should run on a separate thread in order not to adversely affect event handling. The typical sequence is:

1. Implement the run() method to increment a frame.
2. Perform a repaint() operation.
3. Perform a sleep() operation to delay the frame.

Incorporating threading with animation permits the suspension and resumption of animation with button or mouse events.

To suspend or resume animation in an applet when the user leaves the page, it is necessary to reimplement the applet's stop() and start() methods. The following code example shows how to handle a button event:

```
public boolean handleEvent(Event e) {
     if (e.id == Event.ACTION_EVENT) {
        if (e.target == start)
           start();
        else
        if (target == stop)
```

Animation should run on a separate thread in order not to adversely affect event handling.

The JDK provides a versatile and powerful collection of packages to facilitate writing Java programs with active content.

```
            stop();
        }
        return super.handleEvent(e);
    }

    private Thread thread = null;
    public void start() {
        if (thread == null) {
            thread = new Thread(this);
            start.disable();
            stop.enable();
        }
    }

    public void stop() {
        if ((thread != null) && thread.isAlive())
            thread.stop();
        thread = null;
        start.enable();
        stop.disable();
    }
```

A common problem with animation is "flashing," which manifests itself as animation with jitter. One solution to this is to use "double buffering." This involves performing multiple graphics operations on an undisplayed graphics buffer and then displaying the completed image on the screen. Besides preventing incomplete images from being drawn to the screen, double buffering improves drawing performance. This is because drawing to an offscreen image is more efficient than drawing to the screen.

Animating Images

It is likely that loading images will take a long time regardless of whether MediaTracker is used or not. In other words, whenever you load an image using a URL, it will be time-consuming. Most of the time is taken up by initiating HTTP connections. Each image file requires a separate HTTP connection, and each connection can take several seconds to initiate.

One way to avoid this performance degradation is to include multiple images in a single file. Performance can be further improved by using some sort of compression scheme, especially one that is designed for moving images. A simple way to do this is to create an image strip, which is a file that contains several images in a row. To draw an image from the strip it is necessary to first set the size of one image, then perform a draw operation on the image strip shifted to the left so that only the image desired appears within the clipping area.

Conclusion

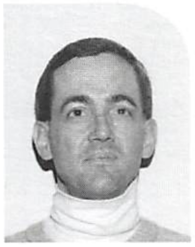
Writing Java programs that contain active content can be very frustrating but also very rewarding. While it may be simple to rapidly prototype an applet with animation and images, fine-tuning it to meet stringent performance requirements requires more than a cursory knowledge of Java.

The JDK provides a versatile and powerful collection of packages to facilitate writing Java programs with active content. The common theme is always to work with the current graphics context and to use classes such as the FontMetrics class to determine sizing of text regions.

java performance

A Cost Model for Java Performance

by Glen McCluskey



Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

In his book *Programming Pearls*, Jon Bentley discusses the use of cost models in connection with programming languages. A cost model identifies the relative execution times of specific language constructs. For example, you can determine roughly how much time is required to perform a divide operation as compared to an add operation, or find out how expensive function calls are. In this column we'll look at a simple cost model for the Java language.

Determining the cost of language features does have some pitfalls. One is that a compiler optimizer may optimize away simple usage, for example, summing two numbers millions of times in a loop.

Related to this problem is the issue of Java just-in-time compilers that perform dynamic compilation and optimization. Such a compiler might, for example, dynamically inline a function.

This analysis uses Sun's Java Development Kit Java interpreter, without optimization and with a switch specified that disables just-in-time compilation:

```
java -Djava.compiler=NONE file
```

where file.class exists and was created from file.java.

This discussion includes a series of benchmark programs and concludes with a summary of results. Download the benchmarks from *<ftp://ftp.glenmccl.com/pub/free/cost.zip>*.

Empty Loops and Arithmetic

The initial set of benchmarks focuses on the cost of looping and arithmetic. The first program measures the cost of an empty loop:

```
public class cost_loop {
    static final int N = 70000000;
    public static void main(String args[]) {
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            ;
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_loop: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

In these programs, the value of N is adjusted so that each program takes approximately the same amount of time to run. Elapsed time is computed in milliseconds and then scaled up to nanoseconds.

The next three programs measure the cost of integer addition, integer division, and floating-point division:

```
public class cost_add {
    static final int N = 50000000;
    public static void main(String args[]) {
        int a = 37;
```

The Java language features several kinds of method calls.

```
        int b = 47;
        int c;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            c = a + b;
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_add: ");
        System.out.println(elapsed * 1000000 / N);
    }
}

public class cost_divide {
    static final int N = 32000000;
    public static void main(String args[]) {
        int a = 659;
        int b = 47;
        int c;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            c = a / b;
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_divide: ");
        System.out.println(elapsed * 1000000 / N);
    }
}

public class cost_fp {
    static final int N = 20000000;
    public static void main(String args[]) {
        double a = 659.84;
        double b = 47.37;
        double c;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            c = a / b;
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_fp: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

Method Calls

The next set of programs measures the costs of method invocation. The Java language features several kinds of method calls. The first one is a static call, a call of a class method without reference to a particular object:

```
public class cost_static {
    static final int N = 26000000;
    static void f() {}
    public static void main(String args[]) {
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            f();
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_static: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

The second is an instance method call. It requires more work because the actual method to be called must be determined dynamically, based on the type of the object. (This type of call often goes by the name "virtual function.")

```
public class cost_virtual {
    static final int N = 24000000;
    void f() {}
    public static void main(String args[]) {
        cost_virtual obj = new cost_virtual();
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            obj.f();
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_virtual: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

The last type of method call is a call through an interface reference, which requires even more work:

```
interface A {
    public void f();
}

public class cost_interface implements A {
    static final int N = 17500000;
    public void f() {}
    public static void main(String args[]) {
        A obj = new cost_interface();
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            obj.f();
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_interface: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

Array Indexing

The Java language specifies that all array subscripts are range-checked dynamically, with an exception thrown for out-of-bounds indices. So it's interesting to examine the cost of array lookup:

```
public class cost_array {
    static final int N = 40000000;
    public static void main(String args[]) {
        int vec[] = new int[10];
        for (int i = 0; i < 10; i++)
            vec[i] = i;
        int x;
        int j = 5;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            x = vec[j];
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_array: ");
```

The Java language specifies that all array subscripts are range-checked dynamically, with an exception thrown for out-of-bounds indices.

Java exception handling is a mechanism for raising, propagating, and trapping exceptions that represent error conditions.

```
            System.out.println(elapsed * 1000000 / N);
        }
    }
```

String Operations

The Java language has a built-in operator, +, for concatenating strings. Java strings are instances of the java.lang.String class and are more sophisticated (and expensive) than simpler C-style strings. Here is a program that measures string operation time:

```
public class cost_string {
    static final int N = 360000;
    public static void main(String args[]) {
        String a = "abc";
        String b = "def";
        String c;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            c = a + b;
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_string: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

Exception Handling and Synchronized Statements

Java exception handling is a mechanism for raising, propagating, and trapping exceptions that represent error conditions. Exception handling tends to be relatively costly, given the work in unwinding the stack, trying various exception handlers, and so on. A program that evaluates this feature is:

```
public class cost_eh {
    static final int N = 18000000;
    public static void main(String args[]) {
        Throwable exc = new Throwable();
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++) {
            try {
                throw exc;
            }
            catch (Throwable e) {
            }
        }
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_eh: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

A synchronized statement – a statement executed after obtaining a lock on an object – is used in thread programming. This program times such synchronization:

```
public class cost_sync {
    static final int N = 13000000;
    public static void main(String args[]) {
        Object obj = new Object();
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
```

```
            synchronized (obj) {}
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_sync: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

Casting

A Java class reference refers to objects of a given class type and also can refer to objects of subclasses of the type. If B is a subclass of A, then an A reference can refer to A objects or to B objects. If I have an A reference, I can cast it to a B reference, but the cast must be dynamically checked, because the A reference might really refer to an A object, which cannot be cast to a B.

Here's a program that measures the cost of casting:

```
class A {}
class B extends A {}

public class cost_cast {
    static final int N = 30000000;
    public static void main(String args[]) {
        A aref = new B();
        B bref;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            bref = (B)aref;
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_cast: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

Creating New Objects

The final area we'll examine is the cost of creating new objects and calling the constructor for each created instance. This particular benchmark also includes the cost of garbage collection, given that you don't explicitly deallocate no-longer-used objects:

```
class A {
    A() {}
}

public class cost_new {
    static final int N = 6000000;
    public static void main(String args[]) {
        A aref;
        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            aref = new A();
        long elapsed = System.currentTimeMillis() - start;
        System.out.print("cost_new: ");
        System.out.println(elapsed * 1000000 / N);
    }
}
```

Creating a new object is more than 40 times as expensive as a simple operation like addition.

Timing Results

We can divide the timing results into sections, according to the above organization. The times include the "overhead" cost of loop iteration, which is around 138 nanoseconds per iteration. The numbers in parentheses are adjusted nanosecond time values, with the 138 ns loop cost subtracted from the raw value in each case.

```
Empty Loops and Arithmetic
    cost_loop: 138 (0)
    cost_add: 175 (37)
    cost_divide: 290 (152)
    cost_fp: 485 (347)

Method Calls
    cost_static: 350 (212)
    cost_virtual: 370 (232)
    cost_interface: 542 (404)


Array Indexing
    cost_array: 192 (54)

String Operations
    cost_string: 25119 (24981)

Exception Handling and Synchronized Statements
    cost_eh: 545 (407)
    cost_sync: 721 (583)

Casting
    cost_cast: 299 (161)

Creating New Objects
    cost_new: 1704 (1566)
```

Using the adjusted numbers, we can make some observations about the costs of various operations. For example, calling a method through an interface reference is about twice as costly as through a normal object reference. Integer division costs about four times as much as integer addition, and floating-point division costs about nine times as much.

Creating a new object is more than 40 times as expensive as a simple operation like addition. The most expensive operation of all is string concatenation, which involves multiple method calls, creation of a new string, copying, and so on.

Summary

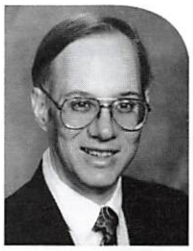
Cost models are a useful way to gain performance insights about the features of a programming language so that you can better gauge the relative costs of different language features.

the tclsh spot

Tcl/Tk is a nice little language, but nothing particularly special by itself. What makes Tcl/Tk exceptional is how easily it can be extended. The easy-to-extend nature of the Tcl interpreter provides a playground for experimentation and has produced lots of useful extensions.

by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.



<clif@cflynt.com>

There are two types of Tcl extensions, ones that add totally new functionality to Tcl/Tk (as the sybtcl and oratcl extensions add database commands to Tcl) and ones that enhance existing Tcl/Tk functionality. These extensions give the Tcl community a chance to play with new ideas, and then pester the folks at Scriptics to roll the best ones into the Tcl kernel.

One of my favorite extensions is BLT, written by George Howlett. This extension enhances the Tcl base functionality with a new geometry manager, a new data type, some enhancements to the canvas, and several new widgets, including graphing and tree display widgets.

Some of the more generic concepts introduced in BLT (the table-driven geometry manager, for instance) have been merged into the Tcl core. Others, such as the graph widget, are sufficiently special-purpose that it makes more sense to leave them in an extension and load that code only when you need to use it.

Which brings us back to the HTTP stock robot from the last "Tclsh Spot" article, and some do-it-yourself historical-data viewing. The previous articles developed a little robot that will query newsalert.com and get prices and volume for a list of stocks. This article will discuss saving and graphing the data the robot returns.

The robot code looks like this:

```
package require http

set formatString {%-5s %-9s %-7s %-5s %-6s %-9s %-9s %-9s %-8s}
puts [format  $formatString \
     symb last change pct date open high low volume]

foreach symbol $argv {
          set url "http://www.newsalert.com/free/stocknews?Symbol=$symbol"

          set id [::http::geturl $url]
          set page [::http::data $id]

          regexp -expanded "arts.Symbol=(.+?)\">   # Get the symbol
                              (.+?)<tr>"          $page m symb data
          regsub -all {<.+?>} $data {} lines
          set dataList [split [string trim $lines] \n]

          puts [eval format \
              [list $formatString] \
              $symbol $dataList]
}
```

To do some historical analysis, we need to collect some history. The simple way to do this is to save data as we read it, which just involves opening a file and writing data.

Tcl I/O commands follow the familiar convention of creating a handle to access the data stream. This handle (called a channel in Tcl) may be used to access a file, device, pipe to another application, or a socket. A channel to a file, device, or pipe is created with the open command.

History without dates is even worse than Friday nights without dates.

This is the syntax for the open command:

Syntax: open *streamName ?access? ?permissions?*

streamName	By default, the name of a file to open. If the first character of the *streamName* is a pipe symbol (|), then the rest of the name is a program to run attached to a pipe.
?access?	The access method: "r" for read, "w" for write, "a" for append. Or a list of POSIX mnemonics including RDONLY WRONLY RDWR APPEND CREAT EXCL NOCTTY NONBLOCK TRUNC. The default is "r" (RDONLY).
?permissions?	When a file is created, you can declare the permissions mask in numeric form. Tcl supports octal numbers, allowing you to set the modes to values like 0666.

We can use a+ for the access parameter, to either create a new file or add data to an existing file.

This command, placed before the loop, will open an output channel for the data :

```
set outfile [open "stock.data" "a+"]
```

The puts command can be used to send data either to the stdout device (as the robot does with the formatted data) or to a specific channel (if the first argument is a channel identifier).

Putting this line inside the loop will write data to the historical data file:

```
puts $outfile "$symbol $dataList"
```

This will generate lines like this:

```
SUNW {75 1/16} {-1 7/8} -2.4 1/28 {77 1/8} {79 1/2} {73 27/32} 19,016
```

which are almost useful.

History without dates is even worse than Friday nights without dates. All operating systems provide a hook for getting the current time. When Tcl/Tk was ported to the Mac and Windows platforms, one problem that needed to be addressed was the different ways systems represent time and date.

The Sun Tcl development team solved this problem by adapting the time and date commands from TclX to generalize access to the underlying time/date representation. The clock command provides access to the system-specific time/date and can also convert from human-readable format to system-specific format and back. For example, you can use the clock command to retrieve the system-specific representation of a time (seconds since an epoch) or to convert seconds to and from human-readable formats like:

```
Wed Dec 31 19:00:00 EST 1969
```

Syntax: clock *subcommand args*

subcommand	The clock command supports several subcommands including:
seconds	Returns current time and date as a system-dependent integer.
format	Converts a system-dependent integer time to a human-readable format. There are many formatting commands to fine-tune the output.

scan	Converts a human-readable time/date string to a system-dependent integer value.

We can decide that we'll save and view the history data on the same platform, and use the clock seconds command to add a timestamp to the output:

```
puts $outfile "[clock seconds] $symbol $dataList"
```

which will generate lines like this:

```
949199093 SUNW {75 1/16} {-1 7/8} -2.4 1/28 {77 1/8} {79 1/2} {73 27/32}
19,016
```

After this robot has been running for a few months we'll have some historical data and can think about looking at trends. Since this is an article about Tcl, not how to analyze the stock market (empirical evidence indicates you should not take stock advice from me), the analysis will consist of reading the history file and graphing selected stocks.

We can architect this program using one of two patterns:

1. Single Loop

```
instantiate graph
open data file
while data in file {
    read data
    plot data
    }
```

2. Split Loops

```
open data file
while data in file {
    read data into internal structure
    }
instantiate graph
foreach dataset {
    plot data
    }
```

The single-loop pattern looks seductively simple, but it's a trap.

Programs (and especially GUI-based programs) should be architected with modules that require a single type of I/O. The module that reads data from a file should interact only with the file; the module that displays data should only interact with display; and the modules that analyze data should only analyze.

Using a single-loop pattern makes a small program, but the monolithic structure makes it difficult to extract a piece of functionality for another program (like reading the data from the file) and leads to spaghetti-like code that is difficult to maintain. (You would have to work around the graphing code when you want to change the data format.) Thus, this graphing program will read the required data from the history file with one procedure and will display that data in another.

A good design for this program would probably be one function to read the data, one function to format it, and one to plot it, but to make life simpler (and this example smaller), the readData procedure will select data for a single company, and will reduce the data to the two fields we'll be graphing, the time and current price. The procedure

Programs (and especially GUI-based programs) should be architected with modules that require a single type of I/O.

will return the time and price data as two lists, which can be passed to the BLT graph widget.

Since the Tcl channel construct for I/O is so generalized, I prefer to pass a channel handle to a procedure that will be reading data, rather than have the procedure open the channel. This puts a bit more of the project-specific information (what kind of channel I'm reading from) outside the procedure and keeps the procedure a bit more generalized.

Finally, the stock prices are reported as fractions, not decimals, but the graph widget only groks decimal numbers. A simple regular expression will split the price into a whole number, numerator, and denominator, and the expr command can convert that into a decimal value.

The procedure to read in the data looks like this:

```
proc readData {infl symbol} {

    while {[set len [gets $infl line]] >= 0} {

        set id [lindex $line 1]
        if {![string match $id $symbol]} {continue}

        set price [lindex $line 2]
        set time [lindex $line 0]

        # Convert "X Y/Z" or "Y/Z" prices to decimal format

        if {[regexp {([0-9]* +)*([0-9]*)/([0-9]*)} $price m whole num denom]} {
              set price [expr $whole + ($num / $denom.0)]
        }

        lappend priceList $price
        lappend timeList $time
    }
    return [list $priceList $timeList ]
}
```

This procedure can be invoked with something like:

```
set inputFileHandle [open $fileName "r"]
set data [readData $inputFileHandle $symbol]

set price [lindex $data 0]
set time [lindex $data 1]
```

Now that the data has been read and formatted, we can graph it. Which brings us back to the BLT extension. Before we can use the BLT extension, we need to either get a precompiled version of the extension (from the Tcl-Blast CDROM, for instance) or download and build it. You can download the latest BLT source from *<http://www.tcltk.com/blt/>*. To build BLT, unpack the archive, cd to the directory it created (probably blt2.4n), and type:

```
./configure;make
```

If you have root permission you can type make install to complete the installation. Otherwise, you can use the extension where it's built by setting the auto_path variable in your script to point to the BLT library directory with a line like this:

```
lappend auto_path $env(HOME)/blt2.4n/library
```

There are two ways to use extensions with Tcl/Tk. The old technique (which works on all platforms) is to create a new tclsh or wish executable linked with the new extension code. When you invoke the new tclsh or wish executable (probably named bltwish for the BLT extension), it will have all the normal Tcl/Tk commands, as well as the new extension commands.

If your platform supports dynamically linked shared libraries (BSD/OS after Version 4.0, Solaris, FreeBSD, MS Windows, or Linux, for example), the configure script will generate a makefile that will create a .so (or .dll on a Windows platform) file in the src/shared directory.

Extensions built from dynamically linked shared libraries can be loaded into a running wish interpreter with the load command, or loaded as needed with the package require command.

The load command is the easiest to use in a development mode. The syntax is:

Syntax: load *libFile.so*

If you are going to make an extension part of your regular coding, you'll want to install the extension in a normal place (like /usr/local/lib) and let Tcl find it with the package require command.

Syntax: package require *packageName ?revision?*

packageName	The name that the package is identified by. This will probably be the letters between lib and .so for a shared library, but might be any identification string.
?revision?	A number that defines the acceptable revisions. A revision number may be a single integer or a pair of integers. A single integer is interpreted as a major revision number, and a pair of numbers is interpreted as a major and minor release specification.

none: Use the newest revision available.
integer: Use the newest revision in this major revision number.
integer.integer: Use only the revision number specified by the major.minor revision-number pair.

Once we've loaded the BLT package, creating and populating a graph is easy. The command to create a new graph widget is graph.

Syntax: graph *name ?option value?*

name	A name for this graph widget, using the standard Tcl window naming conventions.
?option value?	Option and value pairs to fine-tune the appearance of the graph. The available options include:

-background	The color for the graph background.
-height	The height of the graph widget.
-title	A title for this graph.
-width	The width of the graph widget.

The BLT package loads the new commands into the blt namespace, so we create a graph with a command like:

```
::blt::graph .g -title "Stock Prices for $symbol" -width 500
```

This creates an empty graph widget. As with other Tk widgets, when a graph widget is created, a command with the same name is created. A Tcl script can control the widget using that command. You can think of Tk widgets as objects with visible internal state and a single method.

The next step is to display our data. The BLT graph widget deals with data as graph elements. An element is a set of X and Y data and the options that describe how the data should be displayed.

The BLT graph widget command has several subcommands, including the element command, which lets us define and modify a graph element.

Syntax: widgetName *element create ?option value?*

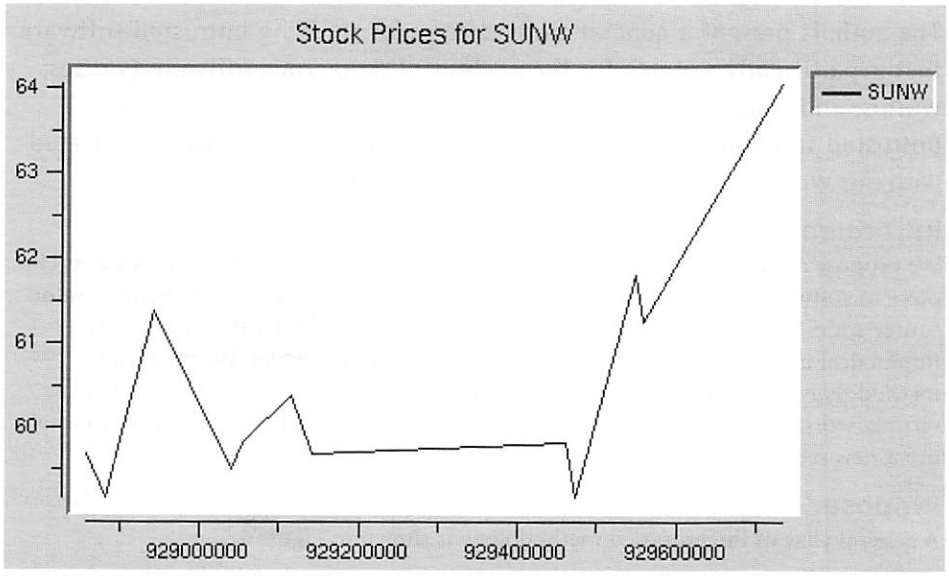
widgetName	The name of a previously created graph widget.
element create	Create a new graph element.
?option value?	Option value pairs that define this element. Options include:

-color	The color of the line.
-symbol	The symbol to display at data points. Values include square, circle, diamond, plus, cross, triangle, none, and user-specified bitmaps.
-xdata	A list of numeric data to display on the X axis.
-ydata	A list of numeric data to display on the Y axis.

Those two commands will create a graph and display a set of data. The complete program looks like this:

```
#!/usr/local/bin/wish8.2

package require BLT

proc readData {infl symbol} {

    while {[set len [gets $infl line]] >= 0} {

        set id [lindex $line 1]
        if {![string match $id $symbol]} {continue}

        set price [lindex $line 2]
        set time [lindex $line 0]

        if {[regexp {([0-9]* +)*([0-9]*)/([0-9]*)} $price m whole num denom]} {
            set price [expr $whole + ($num / $denom.0)]
        }

        lappend priceList $price
        lappend timeList $time
    }
    return [list $priceList $timeList ]
}

set inputFileName [lindex $argv 0]
set symbol [lindex $argv 1]

set infl [open $inputFileName r]
set data [readData $infl $symbol]

set price [lindex $data 0]
set time [lindex $data 1]

::blt::graph .g -title "Stock Prices for $symbol" -width 500
grid .g

.g element create $symbol -xdata $time -ydata $price \
            -symbol none
```

which will generate this image:



when invoked as:

```
graph.tcl stockdata.dat SUNW
```

This is better than nothing, but the dates are in seconds since the epoch, which I don't translate to year and day in my head, and I'd like to see more than one stock at a time. The next article will discuss ways to make this graph prettier and more useful.

a methodology for studying untrusted software

With Application to the Real Case of the WinSatan Trojan

by J.C Hernández Castro, J.M. Sierra Cámara, A. Ribagorda Garnacho, B. Ramos Álvarez, and A. Muñoz Cuenca

The IT Security Laboratory from the Carlos III University of Madrid was created in 1990, and since these days it has been focused on IT security research and development. Its members have participated in multiple research projects and have worked with a number of companies. They can be reached at <*http://benjusuf.uc3m.es/miembros*>.

The authors present a general methodology for studying untrusted software that is particularly suitable for the analysis of dangerous software (viruses, worms, and trojans). In many cases, it is useful to learn exactly what untrusted software does on a computer. This methodology is demonstrated with our work on the recent trojan called WinSatan.

Introduction

We propose a general methodology to extract all the valuable information for a certain piece of software for which we have no initial information, such as documentation or source code. We have found this methodology to be much better than the nearly impractical method of reverse-engineering it with a disassembler. We think this methodology will be of great help in the analysis of potentially dangerous code like viruses, worms, and trojans and could also be used as a reference to develop and implement new security software.

Proposed Methodology

A general view of the proposed methodology is shown in Figure 1.

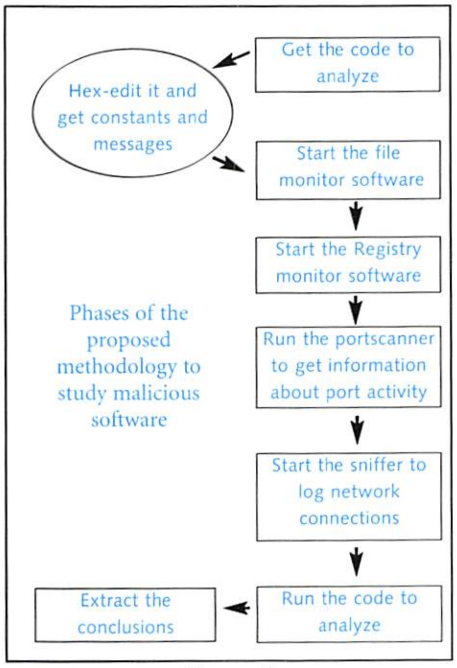

Get the code to analyze
Hex-edit it and get constants and messages
Start the file monitor software
Start the Registry monitor software
Phases of the proposed methodology to study malicious software
Run the portscanner to get information about port activity
Start the sniffer to log network connections
Run the code to analyze
Extract the conclusions

Figure 1

Our methodology is divided into five main steps. Before installing and executing the suspicious software, we recommend the following actions:

1. Use a hex editor (for example, the free HexWorkShop) to navigate through the contents of the untrusted software (usually an executable). In most cases, you'll be able to see all the text messages the software generates, along with some of the values of its internal constants.

When analyzing the WinSatan trojan, we discovered a large number of messages. Some of them were error messages; more important, some included a strange list of active IRC servers and their respective IPs (presumably included to make the trojan robust in case of badly configured DNS servers). This yielded information about the internal architecture and the main features of the trojan. We also discovered that the trojan was programmed using the Delphi programming language, that its author was probably German, and that he/she used the so-called ICL communication components.

2. Use a file-activity-control utility, such as the excellent and completely free FileMon, to follow the trace of all the activity the untrusted software generates during its execution. This generally yields excellent information about files created, modified, and read in order to gather (and possibly publish) information about the victim's machine.

We determined which files WinSatan created upon execution, including the file called fs-backup.exe (which was the real trojan) in the C:/windows directory. We also found that the trojan created a new directory and consulted several other files to gather information. The FileMon utility generated more information than we are able to analyze in detail in this short article. We strongly recommend exiting any other application during the use of FileMon, to minimize the amount of uninteresting file activity logged.

3. Use a Registry-activity-control utility, such as the free RegMon program, to control all the changes and readings the trojan makes in the Registry. This could be of great interest, as most of the known trojans (along with other malicious software) use the Registry as a source of invaluable information about the attacked machine, and they usually create or modify some Registry keys to, for example, force its running every time the user starts Windows.

When we used the RegMon program to study WinSatan, we discovered that it reads some 20 registry keys, but the most important fact we discovered in this phase was that it creates a new one under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run with the apparently innocent and useful name of RegisterServiceBackup and the value "fs-backup.exe". What is this for? Simply to force Windows to execute fs-backup.exe every time it starts up, thus ensuring that the trojan will always be running if the machine is in use.

4. Use a portscanner such as TCPView or the netstat command to scan localhost to observe which ports are open before and after running the untrusted software. In most cases, upon the execution of trojans, new ports are opened on the victim's machine so that the attacker can access it in some way (via ftp, telnet, or some other protocol). We used nmap to scan our infected machine from outside and the netstat command to do it locally. The outside scan is important because the untrusted software (like some rootkits) can modify the netstat command to hide its actions. In both cases, we discovered that port 999 was opened every time we ran WinSatan. Before this discovery, we successfully tried to connect to port 999 and saw that it was an ftp server that allowed anyone full access to the victim's files. The netstat command also showed us that our infected machine was trying to connect to some of the IRC servers we saw in the hex-editing phase.

In most cases, upon the execution of trojans, new ports are opened on the victim's machine so that the attacker can access it in some way.

5. Use a sniffer such as sniffit or windump to sniff all incoming and outgoing traffic. Almost by definition, a trojan will try to open ports on the victim's machine to allow the attacker to take control of it. These incoming connections will be detected and logged if you run a sniffer that logs all incoming and outgoing traffic. Also, the victim's machine usually reports its state to the attackers by some means (e.g., electronic mail, IRC messages, or news postings) and this will be detected and logged if you sniff outgoing traffic.

All this information provides clues to the identity of the author of the malicious software. Using the well-known sniffit sniffer, we observed surprising results: the infected machine sent the following messages to the IRC servers by a privmesg IRC command: "Online!. I am <user or machine name> I use Windows <windows version>, my CPU is a <type of processor>". We had already seen this string in the hex-edit phase, but only at this point did we understand its use. We also saw that this message was directed to two IRC nicknames that were scroll and scroll1 (we suppose scroll1 was created by the developer of this trojan in case some other IRC user had already taken the scroll nickname) and was repeated once every minute to confirm that the victim was still connected to the Net. We have checked that both users are usually active on these IRC servers.

How Did WinSatan Spread ?

WinSatan was described in the news and on some Web pages (for example, <*http://music.acmecity.com/orchestra/29/WinSATAN.zip*>, now gone, where we found it) as "A Windows port for SATAN, the security checker tool for UN*X." This sounds attractive enough to get a great number of downloads and assure a quick spread of the

WinSatan connects to various IRC servers, and this connection remains active even when the program is closed.

trojan over the Net. We discovered that behind this marketing slogan was a dangerous backdoor application.

Technical Details

The trojan used WinSatan to spread. However, none of the software's three functions worked properly, which provably means its only purpose was to spread the trojan.

WinSatan connects to various IRC servers, and this connection remains active even when the program is closed, continuing to run in the background without a trace on the system tray or task manager.

The "I'm online" IRC message was sent by a Privmsg command to the two IRC users, which prevents other users from reading it. Obviously, this exposes the victim's computer to every attack the author of this trojan wants to do on the IRC.

In addition, we discovered that the trojan opens an unprotected (it asks for username and password, but any combination of the two will be accepted) ftp server on port 999 of the victim's machine, which gives the attacker full control over the victim's files. The attacker's strategy is quite simple: he/she only has to connect to one of these IRC servers, using the scroll or scroll1 nickname, and wait for the victims' messages indicating they are infected and online. On receiving one of these messages, he/she only has to discover the victim's IP (trivial, using the IRC commands /who and /dns) and connect to it by an ftp command directed to port 999.

Searching the code with the help of a hex editor, we discovered that the trojan was written in Delphi and that it has a list of IRC servers to connect to. Here is the list (in no particular order):

irc.stealth.net
irc.webbernet.net
ircnet.sprynet.org
irc.univ-lyon.fr
irc.rus.uni.stuttgart.de
eu.ircnet.org
us.ircnet.org
web.im.tut.fi

Windows 3.x, Windows 95, and Windows 98 all proved vulnerable. Tested on a Windows NT 4.0 box, it didn't work at all. (Its 16-bit code on NT gives the error, "The procedure entry point RegisterServiceProcess could not be located in the dynamic link library kernel32.dll.")

To recapitulate: The trojan adds a key in the Registry to run itself. It then runs on every system startup. It tries to connect to the IRC servers every few seconds once the user has connected to the Net. When the trojan manages to connect to an IRC server, it sends the above-mentioned message once a minute.

How to Check Whether Your Box Is Infected

To determine whether the trojan is running on your machine, type the command netstat -an from the command prompt. If you see you are connected to an IRC server using destination ports like 6666 or 6667, for example, 165.121.1.47:6667, and you know you're not running an IRC client, you're in trouble.

Another test to perform is to check whether you have port 999 open. Port 999 is not associated with any standard service, so it doesn't have to be open. If you find it open, try to connect to it with the command ftp localhost 999. If you manage to do it and are prompted with the message "CreaKer is here," you are infected.

Also, check if you have a program called fs-backup.exe (about 366KB) in the C:\windows directory. If you find it, you are infected. Remove it immediately. (If you can't because it is running, simply do a shutdown to MS-DOS and remove it from there.)

In any case, check the Registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

to search for unusual or strange entries. If you find one called RegisterServiceBackup, you are infected. This is a good test to do anyway, in order to catch other trojans that may be lurking on your machine.

Disinfecting

Disinfecting a machine couldn't simpler. Just remove the Registry key that allows the trojan to start when Windows starts (named RegisterServiceBackup, as described above). Restart Windows, and, since the trojan is not running, you will have no problems removing the file fs-backup.exe from C:\windows. Remove it and you are clean!

WinSatan Today

We tested some well-known antivirus programs to see whether they could detect the new trojan. None of them did, even those using heuristic methods. We tried McAfee VirusScan version 4.0.3 with heuristic searching and the latest update by this date (6 June 1999), Antiviral Toolkit Pro 3.01.29 with the latest revision (9 June 1999), and the Platinum Panda Antivirus 6.0 with heuristics after an update on the Internet. We emailed McAfee, AVP, and Panda, and all of them included the code signatures of the trojan in the next versions of their products.

The discovery and analysis of this trojan has been widely covered, mostly by the digital media. As a result, many people have contributed in different ways to avoid its spreading but, unfortunately, a lot of people have been infected by it,.

Thanks to Leo Ricchieri, who has developed a VB script that removes the trojan; see *<http://www.securiteam.com/securitynews/WinSatan_FixBackdoor.vbs>*. (This requires a VB script–enabled browser.)

WinSatan's characteristic port 999 has been included in many default trojans ports lists all over the Net. Last, but not least, one of the discoverers of the trojan has developed a Nessus script that detects WinSatan remotely, so any user of the excellent and free Nessus security-auditing tool can detect whether a remote machine is infected by the WinSatan trojan. More information on Nessus can be found at *<http://www.nessus.org>*.

Unfortunately, in spite of the efforts made by many people to stop the trojan's spread, many users are infected, and many crackers are abusing them. The nicknames scroll and scroll1 on the cited IRC servers are almost always used by people abusing victims.

Conclusions

The proposed methodology proved completely adequate to analyze the WinSatan trojan. We managed to discover most of its features very quickly and to publish an alarm report that has been widely distributed on the Net. The authors have used this same methodology in the analysis of other trojans (Back Orifice 2000, NetBus, etc.) and have found it works pretty well, allowing us to discover the trojan's structure. We think, however, that the WinSatan trojan was a better test of this methodology, because it was discovered in the wild, with no information available, whereas Back Orifice 2000 and NetBus were widely documented. One of the main advantages of the proposed methodology is that all the tools are freeware, so it is available to anyone interested in malicious-software research.

BIBLIOGRAPHY

Denning, Peter J., ed. *Computers Under Attack: Intruders, Worms, and Viruses*. New York: ACM Press; Reading, Massachusetts: Addison-Wesley, 1990.

Ferbrache, David. *A Pathology of Computer Viruses*. London: Springer, 1992.

Hoffman, Lance J., ed. *Rogue Programs: Viruses, Worms and Trojan Horses*. New York: Van Nostrand Reinhold, 1990.

Ludwig, Mark A. *The Little Black Book of Computer Viruses*. Tucson: American Eagle Publications, 1991.

the network police blotter

by Marcus J. Ranum

Marcus J. Ranum is CTO of Network Flight Recorder, Inc. He's the author of several security products and of a book on computer security (with Dan Geer and Avi Rubin) and is a part-time sysadmin.

<mjr@nfr.net>

Some Results

Mark Maris sent in a really nice haiku:

The extra bits fall
out onto the command line . . .
buffer overflow!

And another good one from Jason Testart:

On the bastion host,
a flurry of port scanning.
No process listens.

Hugh Kennedy gets honorable mention:

root password mumbled
aloud in a crowded lab.
how fast can i type?

I'm sending each of these worthy gentlemen a cool T-shirt that will make them the envy of their friends. I'm going to see if I can get them and the other contestants to let me publish their efforts on the Web. By the time you're reading this, I should have them online at <*http://pubweb.nfr.net/~mjr/usenix/haiku*>.

Building Burglar Alarms

My first "real" job was working for a burglar-alarm company when I was a kid in high school. Now, umpty-whatever years later, I'm basically doing the same thing for a living again. My boss at the time used to design great alarm systems; a lot of historic properties and municipal buildings in Maryland (including a nuclear-power plant) had alarm systems that had been designed specifically for those sites by my boss.

One of the distinguishing characteristics of his alarm systems was that he liked to put special traps within the perimeter of the site that was being secured – not just at the boundary of the network, um, uh, building. So the alarms had all of the usual glass-break detectors, window switches, and so on – with the addition of window switches hidden in gun cabinets, jewelry boxes, executive desks, key boxes, and so on. When we wired an alarm system with a bell, we'd put in a low-power circuit that would cause the alarm to go off (silently, of course!) if the bell circuit was cut. If the bell was in a box on the outside of the building, we'd include a contact switch in the bell box to set off the alarm if it was opened or pulled away from the wall.

Never mind infrared and microwave motion detectors (which are pretty obvious LCD-blinking white boxes stuck to a wall) – we used to put pressure-sensitive pads under carpeted floors in hallways. I guess I learned paranoia at an early age! Remember, one person's "paranoia" is another person's "engineering redundancy" – these traps worked extremely well. The special alarms would trigger a different code from the normal ones, so if the system called the police, the cops would know that they were dealing with a professional who had bypassed the first layer of security.

Obviously, you can see where I'm going with this: the same concepts apply to network and even application security.

If you've got a system that you want to keep people from breaking into, ask yourself a couple of leading questions:

- What am I really afraid could go wrong with my system?
- What would it look like if something were in the process of going wrong?
- What can I put in place to tell me when it is happening?

That's the simple recipe for a decent computer burglar-alarm system. Put alarms in place and I guarantee you eventually you'll come out looking like a hero – someone will break into your system using some ultra-sophisticated new trick, and you'll find out about it virtually instantly. While everyone else is running around in panic and confusion, you'll be the person on the spot armed with knowledge; it's a nice feeling!

The only drawback with burglar alarms is that we can't all use the same burglar alarms and have them work. To work correctly, the alarm has to be a surprise for the bad guy. Think of it as scattering land mines around your network. If we all put them in the same places, laid out in a nice, standard, neat grid, then the bad guys will instantly know where not to walk. So you need to be creative and think up your own burglar alarms. As tasteless as it seems, the land-mine analogy is actually very good because, like mines, burglar alarms are probabilistic devices. The more mines you put in a given area, the greater the likelihood someone will set one off if they enter that space. The more traps you put on your systems, the greater the likelihood someone will set one off if they enter your computer.

A Few Good Tricks

Most firewalls let you "overload" rules in front of one another. By placing a more specific rule "ahead" of general rules, you can cause the firewall to take different actions even against traffic within your own network. Suppose you have a Web server behind a screening router that is your boundary to the Internet. Consider placing screening rules in the router that "overload" the generic outgoing rules, so that if the Web server starts generating traffic to the outside that it normally wouldn't (say, IRC or ftp) it generates a log message. The rest of your network traffic gets different rules applied. Obviously, this only works if your Web server produces and consumes only a limited number of services. If it's running a zillion servers on various ports, then you may as well forget about security on it, anyhow.

Or suppose you have a Web server on a "DMZ" (semi-public) network outside of your firewall. Consider putting rules in your firewall that block and log traffic from the Web server trying to come in to your network. Give those rules higher precedence than the normal "block everything coming in" Internet rules. That way, it's a dead giveaway if someone breaks into your Web server and tries to probe for a way into your internal network. Consider installing in-kernel firewalls on perimeter machines such as Web servers or external DNS servers. Overload the firewall rules so that they generate alarms if the other systems on your DMZ network start probing one another. Have them generate alarms if they get traffic that should have been screened by your boundary router. Since none of these things should happen, the alarms should never fire. Right? You might be surprised some day.

Another common thing bad guys do when they get on your system is remove logs.

Another common thing bad guys do when they get on your system is remove logs. If you're at all familiar with using make to build C programs, it's very easy to replace or add useful traps to your existing system software. For example, a fun thing to do would be to modify a long-running daemon so that it monitors the system log file and raises an alarm if the file ever vanishes or gets shorter during the day. Just hold an open descriptor to the file, and if the inode number returned by fstat is not the same as the inode number returned by stat, and/or the file size is different, then you know you have a problem. If you've still got the descriptor open and the bad guy forgot to truncate the

"Are you trying to say that we should all quit trying to secure our networks and get jobs as plumbers or something?"

file and simply removed it, then you can lseek back to the beginning and write the file out somewhere safe. Essentially, you're trojan-horsing your own system – except that when *you* do it, it's legal and ethical.

If you're running a dedicated system that isn't for general-purpose use, you can replace system binaries with ones that raise an alarm if they are run. An extreme case would be replacing ls and more with programs that raise an alarm if they are ever run with privileges. Then train yourself to use echo with shell wildcard expansion, or some other means of seeing what's on your system. If you're running a multiuser or guest system, and you're not afraid of intruding, consider replacing networking programs such as telnet, ftp, and IRC with versions that log who ran them and to what they connected. You may wish to modify your kernel (if you're into that) to record stuff directly from the connect system call, as well as to record whenever the execute bit is set on a file. I recognize and respect the privacy issues such actions would raise; these ideas are not for everyone.

A few more ideas: many of the free UNIXes let you set an option to generate an error if something tries to execute code off the stack. There are also software solutions for detecting stack-smashing attacks from within the runtime environment. Consider turning such tools on, if you're managing a Web server, and you may be the first (or, actually, the second . . .) to know of the latest buffer-overrun attack. If you've got the time, it's not too hard to have a program monitor the process table and watch your Web server to see if anything weird happens to the process. Weird things that shouldn't happen: key processes vanishing, getting smaller, dramatically changing the amount of CPU they use, etc. You want to know these things, believe me! You probably don't want to have a process sitting there monitoring your process table – it'll stick out like a sore thumb. There are all kinds of fun places you can put it: as a subroutine of inetd, cron, or even sendmail, to name just a few.

Web-site defacement was all the rage in the press last year. It's still a big issue if it's your site that gets hit. Make sure you're the first person to know about it, if it happens. If you've got a friend on the outside with a high-speed connection, set up mutual watch scripts on your sites that monitor each other's root page (or pages) for changes. There are also Web-page-watching services that can be set up to email you if a page changes; have it watch your page and mail to an alphanumeric pager. That's probably not as fast, in terms of alert latency, as it should be, but it's cheap and easy.

The Plumber's Tale

I was giving a talk at a conference recently, and I'm afraid I was being a bit depressing and slightly cynical about the state of Internet security.

One of the people in the room asked a really thought-provoking general question, which was, "Are you trying to say that we should all quit trying to secure our networks and get jobs as plumbers or something?"

I have to admit that I was kind of speechless for a second, and mumbled something about never giving up, never surrendering, and so forth. It wasn't until later when I was sitting around with some friends that we collectively thought of a much better answer, namely that being a plumber isn't much of a job change.

As security or firewall administrators, we've got basically the same concerns: the size of the pipe, the contents of the pipe, making sure the correct traffic is in the correct pipes, and keeping the pipes from splitting and leaking all over the place. Of course, like plumbers, when the pipes do leak, we're the ones responsible for cleaning up the mess, and we're the ones who come up smelling awful . . .

Next Up

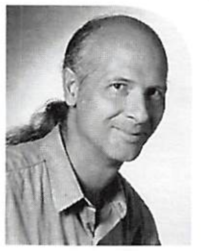
Did a few of the dirty tricks I proposed above inspire you? Send me your favorite(s), and if yours is the best, I'll send you a totally cool Network Police windbreaker. Send them to me <*mjr@nfr.net*> with a subject line reading "dirty trick" and I'll sort through them for my column after the next one.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*

<rik@spirit.com>



Dare we celebrate? I think not. Just because Microsoft has lost its browser monopoly case doesn't mean that anything will change soon. Microsoft employs many of the most brilliant programmers and marketeers this world has, and some of the things they do are really amazing. Yet Microsoft will continue to wield its 800-pound gorilla image, seeking to impose its will "in the good of the people," at least until the appeals process winds down years from now.

I did buy a Sony VAIO 505VE, a tiny subnotebook that I had seen in the hands of a Japanese man at a USENIX conference years ago (it seems). I immediately set about installing a version of Linux (Red Hat 6.1), because I wanted the notebook for several purposes that were not aligned with the version of BSD I run at home. That is, running hacker's tools for demonstration purposes, and running VMware.

The installation went smoothly, once I found the right information at a Web site (*<http://www.cs.utexas.edu/users/kharker/linux-laptop/>*). For example, you must add flags to the boot line or the CD-ROM will not be recognized by the Linux kernel. Getting the X server running was trivial. But getting the internal modem working was extremely frustrating, because it turned out to be impossible.

The internal modem, exposed via a little flap in the back left corner of the subnotebook, was supposed to be a v90 supporting 56k. It turned out to be a winmodem, essentially a modem implemented partially in software that will not run in a real operating system (at least, not yet). I even reinstalled Windows 98 just to test it and see whether I had a hardware problem. Worked fine under Win98, although I had to brandish a big stick several times as Win98 and the Sony stuff kept trying to sign me up for several Internet services.

Linux software, such as wvdialconf, could see that there was a UART installed but could not talk to it. The same held true for a modem card I stuck in the single PC slot. I had checked for interrupt conflicts under /proc and didn't see any. Finally, I checked again at the linux-laptop site and there were several new pages posted about the 505VE. There I learned that it had a winmodem and that I needed to add a line to the pcmcia.conf file to prevent interrupt conflicts. Geeze! The network cards I tried worked like a charm, but not the modem cards (or even the part of a combo card that was the modem).

Once I had given up on using the internal modem, I got a 3COM combo card, 56k modem, and 10/100 MHz LAN (3CCFEM556B). Worked great under Linux, but required me to install software under Win98. That is when the Win98 partition died. It is really amazing to me that installing something (using the standard Install Wizard) would result in Win98 losing critical files, which would in turn require reinstallation. Installing Win98 from the recovery CDs also trashed the Linux partition, and at that point I punted on Windows.

VMware has been working great. I have not yet installed the 2.0 version, but have been running NT4/SP4 so that I can run PowerPoint. Not that I ever wanted to run PowerPoint, mind you. I bought Office 97 so that I could see if the way I configured NT (ACLs and such) would prevent Office from working for non-administrators. I was also curious about Word macros and about the ability of Internet Explorer to execute various malware on a system. The nice thing about VMware is that you can save an entire copy of an NT installation, so if you trash it through testing malware, you can simply run a copy created before the test. Much easier than trying to reinstall.

One thing I did notice about running VMware and IE: I cannot access Microsoft's Web sites. I have no problem with other Web sites, but get only blank pages from Microsoft. (Navigator under BSD or Linux does not have this problem.) Perhaps Microsoft does not like VMware? How do they know I am running VMware?

Also, thanks to Jill Sole of the NSA for hand-entering all 320 pages of my Intrusion Techniques and Countermeasures course. The NSA wanted to use the course internally, and that required that it be a PowerPoint presentation. My first experience with running PowerPoint was a prophetic one – it turned out that about 20% of the slides I had edited were blank! This has happened on random occasions since, but I learned my lesson long ago. I just started treating PowerPoint the way I treated WordStar under CPM back in 1980: you make a backup copy every ten minutes or so. In my case, that meant saving the file and sending a copy via ftp to a UNIX system (keeping redundant backups there, of course). Just amazing how far software has come in 20 years!

If FireWall-1 were a child's toy, it would be recalled as too dangerous.

Worthless Firewalls

Having spent years promoting the benefits of firewalls, I have my own systems sitting behind a firewall. Mostly, the firewall does a good job (is nonintrusive; deletes dangerous content, that is, attachments for Microsoft products, automatically; and works much faster than the max throughput over my WAN link). Occasionally it does get in the way – for example, when I go to TCP fingerprint someone with nmap (*<http://www.insecure.org/nmap/index.html>*). I have to remember to stick a machine outside of the firewall before doing that, or I wind up scanning the inside of the firewall.

On the topic of firewalls: there has been a trend toward what I call the "checkbox firewall." The checkbox firewall is the one where the auditors check to see whether there is a firewall, and then check the appropriate box on the audit sheet. In one case, the firewall was actually still sitting in its original package (have a firewall, check!). In another, the firewall was configured to permit any source, any destination, and any service without logging. Worked fine, although it did tend to slow down the Internet link a little. As a security device it was totally without merit.

One of the leading firewall vendors, CheckPoint, has shipped a firewall for many years that can be configured into something reasonably secure. But the design of the user interface, and the defaults, encourage configuring the firewall insecurely. If FireWall-1 were a child's toy, it would be recalled as too dangerous. It epitomizes what I consider evil as far as design goes. Here's why.

To start out with, the design focuses on speed and flexibility. Neither of these features has anything to do with security. The defaults make the product easier to install, while leaving gaping holes in security. For example, you can scan through FireWall-1 (and many other firewalls) by using TCP packets with the ACK bit set. Tools like nmap make this easy to do. CheckPoint might argue that the firewall is doing NAT (Network Address Translation), making internal addresses not routable, but would this be true from the local ISP? FireWall-1 also leaves ports 53 TCP and UDP unprotected by default. At least later versions include a prepackaged definition for controlling DNS to each transport protocol. Too bad they don't filter out ICMP packets either.

But the real coup de grâce has to do with the interface itself. It permits the naive user to permit any of over 100 network services, most of which should never be permitted through a firewall. Now, none of us would be so unwise as to do that, right? Or only do

TCP/IP, and most of our modern operating systems, were not written with security in mind.

it when the firewall is being used internally to partition networks. But what about most of the people who buy firewalls? What I have found is that if the firewall supports a network protocol, the customer assumes that it must be safe. In other words, the user interface misleads people into thinking that it is safe to permit NFS or SMB through the firewall (or even TFPT and SNMP), when, in truth, these protocols should be run in protected subnets.

When the market leader sets the stage, market followers generally converge on similar designs. This has meant that stateful packet filtering has become the standard for firewalls, even when it is not appropriate. An interesting hack was published in late February whereby you could open holes in SPF firewalls if you had access to an ftp server protected by that firewall.

In the example exploit, an unpatched Solaris system running an ftp server was hacked through FireWall-1 running on a Nokia router. By setting the MTU to 100, connecting to the ftp server, then sending a long string that included "227" at the packet boundary, you could trick the firewall into opening the port occupied by the ttdbserver and use an exploit to copy a shell in place of the ftpd. The "227" is the response sent by an ftp server to indicate that it is ready to receive a passive data connection, something that the firewall must respond to. Application gateways do a much better job at keeping track of state, regardless of the fact that the word "state" appears in SPF.

The Future

Alas, when I look into my crystal ball, the future I see is dark indeed. When firewall vendors, operating-system vendors, and networking vendors all appear bent on assuming that security is not an issue (ease-of-use and performance is what counts), then we are in trouble. It is not the case that doing security correctly is easy. Rather, security must be done right from the beginning, if the infrastructure we are building is meant to be robust. And TCP/IP, and most of our modern operating systems, were not written with security in mind.

Somehow, security must become as important as performance, or we will continually be spending our time cleaning up after intrusions, or just never keep any data that must be kept confidential online. So long, e-commerce!

Years ago, the security product manager at Sun told me that the reason Sun did not deliver its product with secure defaults (remember the plus sign in /etc/hosts.equiv?) was that customers did not include requirements for security when making purchases. In fact, making a system more secure generally makes it more difficult to use. Just plug the system into the network, give it an IP address, and it immediately becomes exploitable. How useful.

I still am foolish enough to get up in front of crowds of people and suggest that diskless workstations running some simple operating system (Java stations?) replace diskful desktops running Windows. Supporting Windows desktops is insecure, hard to maintain, and costly. Some people I mention this to immediately say, "Yes, going back to mainframes makes sense." No! Going back does not make sense. But replacing stupid desktops with simpler ones does make a lot of sense. People can play games at home, on their own PCs, and pay beaucoup bucks to the local Windows expert who knows how to replace the missing DLL that prevents their system from booting.

If anyone wants me to join a board of directors or tech advisory board on diskless workstations, let me know. I still believe this is the wave of the future, and I want to be on the crest before the wave passes.

system administration research

Part 1: System Administration at the Crossroads

The Delicate Art of Assertion

by Mark Burgess

Mark is associate professor at Oslo College and is the author of cfengine and winner of the best paper award at LISA 1998.

<Mark.Burgess@iu.hioslo.no>

I like to think of myself as a scientist. I always wanted to do science; I studied physics and have since interested myself in many other areas, not least among them computers. There is something about the creative pursuit of understanding that appeals to my mindset and motivates what I do. On the other hand, I have always found the world of commerce somewhat bewildering.

Apart from feeling nauseated when I am forced to think about money matters, I have an almost pathological distrust of sales promotion. The funniest things get said in the name of marketing. Wild assertions like "You know it makes sense" and "Eight out of ten owners said their cats preferred it." I recall a version of Solaris being released with the slogan "over a hundred bugs fixed," as though this were a good thing. What were they doing there is the first place? Although I am not so naive as to believe that making money is a bad thing, I do believe that the subjective metaphors of marketing have no place in the objective goals of science.

The field of system administration is now at a crossroads, poised between selling its soul to the devil of marketing (sorry, devils) and finding a more rigorous foundation in the tradition of science (cue angelic choir). Like most engineering disciplines, it will no doubt split and go both ways. It is important that a large part of it takes the road to scientific salvation.

If I haven't already made it plain, I'm on the side of science. This is the side that SAGE has been steadily encouraging since the first LISA conferences, by opening up a forum for vendor-neutral discussion. I would like to see much more scientific, technical work presented at the LISA conferences and far fewer descriptions of tools.

Looking back through all the LISA conferences while writing my new book on system administration *[Editor's note: See this issue's Bookworm column]*, I saw a panel debate very early on that asked: why do we keep reinventing the wheel? Pursuing my journey through the papers to the present day, my only answer was: why indeed? Very little progress has been made in the field in more than ten years. This is mainly characterized by a slight gradient in the level of sophistication, fueled by technological advances.

This set me thinking. Why are we not doing more scientific work? In recent years, it has become popular to study Internet traffic and network behavior in a scientific way, so that the idea of empirical work has become virtually synonymous with testing network behavior. But network behavior is just one tiny part of system administration. What about the rest of it?

This series of articles is aimed at activating the sleeping academic in all of us. The most constructive way, arguably, to complain about the state of the art is to try to improve on it. I would like to provoke that development. The wonderful thing about science is that anyone can do it. You do not need a fancy degree to make a contribution (though

The aim of this series is to explain the meaning of system administration as a scientific pursuit.

the training you get often makes it easier); all you need is a critical sense and a measure of discipline.

Many problems must be faced in addressing system administration scientifically. This is surely why its development has been slow, in spite of the many scientifically trained system administrators in the field. The aim of this series is to explain the meaning of system administration as a scientific pursuit: what is it we should be looking for? How do we look for it? How can we be sure that what we're claiming is correct? How can we be our own worst critics?

Before starting, I would like to offer a word of warning. Like marketing, personal promotion through research competitiveness has no place in science, yet it thrives like a virulent plague among all-too-human researchers, eager to "beat the competition" and be the best by bashing the rest. Since I find LISA's friendly and open-minded atmosphere a refreshing change from many other conferences I attend, I have worried that bringing more critical rigor into the field might also bring the arrogance that frequently attends it. As soon as one person starts being more academic, there could be a race to be the loftiest of them all.

The key is to keep that race constructively aimed at the goal of the research and not at self-promotion, or engaging in unfriendly one-upmanship. Let us try to avoid these specters as long as possible and cooperate rather than compete; they will no doubt haunt us soon enough. That said, down to business.

Critical Sense and Common Sense

So I'm standing in the corridor and a student from my system-administration course says to me, "Why is Solaris server X soooo sloooow? Couldn't we just get an NT server and be done with it?" After asking him to remove his glasses, so I could do the thing with two fingers, he laughed and disappeared, but then a moment of humor faded into depression. How could someone in my course say such a thing?

For a start, the server that was relevant to his work was a GNU/Linux machine, not the Solaris machine to which he referred. Second, a moment's investigation would have shown that this host was running at lightning speed and doing practically nothing, but that something in the network (a bad switch as it turns out) was causing an appalling network-transmission rate. Third, how would running NT make anything faster?

What made this worse was that I have heard comparable abuses from considerably more informed practitioners. For instance, I once heard, "GNU/Linux is at least as fast as Solaris at NFS." This woke me up. What could this possibly mean? Actually I don't care which is faster, but what was the person trying to say? Were the two operating systems ever compared on identical hardware, under exactly the same conditions? Under what range of tests was this investigation carried out? What hardware was involved? Were the networks the same? Were any measurements actually made, or was this a gut feeling? I suspect the latter.

Here's another one: "Bourne shell scripts are lighter on the system than Perl because Perl has a huge binary, whereas the Bourne shell binary is much smaller." In fact, it is possible to settle the issue by measurement. That is the scientific thing to do. If there is a question, one carries out an investigation. No assertions are without risk of being toppled by contrary evidence. As it turns out, that assertion is full of holes. First of all, let us look at the sizes of the binaries on disk:

```
     GNU/Linux
516828 /usr/bin/perl
373176 /bin/bash

     Solaris
718688 /local/bin/perl
91668 /bin/sh
688972 /local/gnu/bin/bash
```

System administration is a young field in academic terms, but there is no excuse for unscientific exclamations in a public arena.

The Perl binary is indeed significantly larger than /bin/sh on the Solaris machine, but not much bigger than bash. The Solaris binaries are generally larger than the Intel ones, since the SPARC processor is a RISC architecture, so it is not even generally true that the Perl binary is bigger. But comparing binaries on disk is far from the point: what is interesting in a system-administration context is how much of the system's resources are consumed by a script. The size of the binary on disk is irrelevant. What we need to know is the total resident size of the program in memory! This is now a complex thing. Consider the RSS measure from ps ux on a GNU/Linux host:

```
924 bash
824 perl
```

Here we find that an actual measurement gives Perl a lower resident size at the time of measurement. Moreover, a shell script is not a complete program – it calls shell commands, each of which is a forked process, adding its own memory imprints to the sum, as well as a lot of context switching and interprocess communication (pipes and the like). Since Perl does not require pipes in order to communicate internally, for most things it suddenly seems like a very lightweight language compared to the shell.

System administration is a young field in academic terms, but there is no excuse for unscientific exclamations in a public arena (behind closed doors, over beers, we can BS all we like). Marketing slogans: my program is more secure than yours because it is written in Perl rather than C, my window system is better than yours because more users have it. These contain no truth, they make little sense, they are hot air. Trusted experts, whose opinions are respected, should not make sloppy and unfounded remarks that are unworthy of them. Never is this more prevalent than in operating-system wars or favorite-software scuffles. X is better than Y because I have a good story to tell. The aim of introducing a more scientific culture to the field is not to pull rank, but precisely to make rank-pulling impossible.

Science

Science is a meld of two ideas: theoretical model building and empirical data gathering. Scientists are hunter-gatherers! These two belong together. Models are needed to interpret empirical data and motivate experiments, and data are needed to substantiate theories or to inspire models.

As an empirical science, system administration leaves a lot to be desired. It is generally not hard to make measurements using the variety of programs available to us; rather, the problem is that in order to make a verifiable assertion, we need repeatability. An experiment that can be repeated many times with the same essential result can be trusted far more than a result obtained only once. It weeds out flukes. But repeatability is virtually unobtainable in social sciences (and system admin is a social science), because when we try to measure things where large groups of people are involved, the conditions under which measurements are made change constantly.

Comparing the results of one trial with the results of another requires performing them both under identical conditions. This problem has confounded the social sciences

Science is not just about measuring stuff, it is also about separating effects that do not belong together.

from their outset, but there the problem lies in asking questions that are too broad or too vague. In system administration, we must ask smaller, more concrete questions.

For instance, it does not make sense to compare the performance of a computer system during the day with performance measured during the night, and then use the average of those values as being the true performance. There are all kinds of things going on in a computer system that contribute to (and cannot be separated from) performance measurements.

What users do is a very significant influence (often the most important influence) on the system. Comparing a measurement at different times during the human social cycle is asking for trouble. During the night computer users work, during the day they sleep. (Or is it the other way around?) If multiple measurements are to be made, they should be made under virtually comparable conditions and then analyzed statistically to take into account small residual variations in conditions of measurement. Science is not just about measuring stuff, it is also about separating effects that do not belong together. This is one of the themes we shall return to in detail in this series.

So much for experiment. As a theoretical science, system administration almost doesn't exist. Actually, as a theoretician, I have been working on this problem, and I'll get back to this in good time. This should not be taken as an indication that theory is unimportant. On the contrary. Real progress will not be made in science without the ability to refer to a model, a set of assumptions, goals, and intentions. This is where theory will enter.

What kinds of issues will be we able to address with research into system administration?

- Evaluating system policies
- General investigative, troubleshooting methods
- Optimal strategies for solving problems
- Elucidating failures and problems
- Discovering the need for new technology
- Comparing technologies
- Predicting problems in advance
- Building simulations of complex systems

Cause and Effect

The central principle behind the behavior of any mechanism is that every effect (every change) is the result of a cause. This might sound painfully obvious, but it is also the part of investigation that is so obvious that it is frequently forgotten by the inexperienced. The art of making an investigation, as Sherlock Holmes knew, is to build a chain (sometimes a more complicated web) connecting the observed evidence with a proposed theory. Until this is done, any "explanation" of a phenomenon is simply speculation or assertion.

- Something happened or changed.
- What changed?
- Does it continue to change?
- Measure the change.
- What processes affect the values you measured?
- Guess an explanation for the change (theory).
- Test the theory by carrying out several experiments.

It is not always possible to prove or disprove results. Usually the world around us is so complex that we cannot know every detail of the causal chain, and uncertainties blur our understanding. This is where statistics come into the picture. If we do not get exactly the same result every time, then we are missing some piece of the puzzle, but maybe it doesn't matter. Maybe we only need to know the causal web approximately, taking into account the main reasons and ignoring the minor changes by calling them "errors" or "uncertainties." In this case we can never actually trace every detail from cause to effect and "prove" a theory. It is only possible to say that something is probably true. The corollary to the above list is this:

- Do the experiments agree exactly? Were there some discrepancies?
- Are the discrepancies as big as the measured effect, or small?
- Is your proposed explanation plausible?
- Look for every possible flaw in your argument.
- Are there alternative explanations?

A brief note on case studies. Case studies are anecdotal evidence, useful for motivating work. Case studies are inherently flawed as scientific evidence, however, because they are never repeatable. They suffer from the sociologists' dilemma of never being able to repeat under the same conditions. Thus case studies can never be used to prove a point, only to suggest an explanation. In many cases case studies may be the best we can do, but we should strive for more.

Case studies are inherently flawed as scientific evidence because they are never repeatable.

Statistics

Statistics is the course I hated most throughout my school and university years. I hated its pompous jargon and its incessant fascination with rolling dice. It was only later, when I chanced upon the meaning of statistics through the back alley of quantum physics, that I realized that there are, in fact, a few key ideas in statistics that my school-day arrogance had refused to let me see. They lie at the core of an opaque and intensely tedious melanoma of self-importance, whose turgid propensity for making simple things sound technical is paralleled only by postmodern philosophy. Whew! There, I said it and it felt really good! (Just for the record, I am laughing.)

Those who manage to see through or bypass the opacity of statistics courses learn to apply the few principles and understand their meaning. Nothing is worse than a bad statistical analysis for making nonsense of data. But a good statistical analysis can result in a significant improvement in understanding data. Of course, that assumes that it is relevant to apply statistical methods at all.

Statistics is really about classifying the different kinds of change that result from a complex web of cause and effect. We resort to statistics only in situations in which details are missing from our understanding of the link between cause and effect. For instance, user creates new file, new file appears. This experiment does not have to be repeated many times before one understands that the identical outcome results from the identical cause. We do not have to collect statistics about this, because the operations are so primitive that we can trace every part of the change and develop a theoretical explanation that details every link in the chain: it has to work.

Of course, the above example also illustrates the point that, on occasion, apparently well-understood phenomena surprise us. If the file system is incapable of creating a new file, that trusted experiment will also fail, and we must seek a new link in the causal chain to explain the result. Collecting statistics would not help us to understand why the act of creating a new file fails on occasion.

Statistics are useful when changes are tangled in a web of intrigue and we are not able to understand all of the influences or the results they give rise to. In that case we can do several things: if there are both big changes (trends) in data and small changes (fluctuations), then these can be separated by averaging. The average separates the large changes from the small ones, because it smudges out small details.

Normally one plots an average value and uses error bars to indicate the actual spread of values that were averaged over. The error bars indicate the size of the small fluctuations, or deviations from the "normal" average. If the size of the error bars (the spread of values) is not much smaller than the size of the effect we are looking for, then we cannot trust our statistical analysis: it tells us nothing meaningful (though it might help us quantify the meaninglessness). Indeed, it tells us that separation of large and small is not possible. If the error bars are small enough, then it means that the main features of the data can be understood in terms of the average values.

What this does is to provide a justification for ignoring a minor source of change (the small fluctuations) that we do not want to deal with, allowing us to focus on the main effect. In other words, statistics is a filter for cleaning up a noisy signal or for classifying different parts of the noise. The techniques of statistical analysis are all variations on this simple theme. Statistics is a useful tool, just as a graphic equalizer is useful in a sound studio, but it is not a guarantee for finding meaning.

An Ongoing Discussion

We shall return to more concrete examples in the remainder of this series. With LISA 2000 approaching, I would like to encourage anyone with an interest in their systems to think long and hard about how they can participate in raising the discussion about system administration to a more scientific level. It is not necessary to present a finished tour de force. That is not the real purpose of a paper. It is more realistic to expect to be able to present work that raises questions that could require several years to answer. Ask yourself:

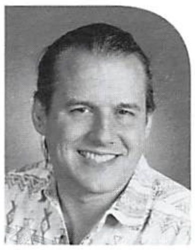
- Do I have something to say? Even something small?
- Will this advance the state of the field?
- Will this contribute to a larger discussion?
- Is it about the behavior of hosts?
- Is it about the behavior of users?
- Is it general, or is it a specific example?
- Can others learn from my experiences?
- Have I thought of every possible explanation?
- Have I explained how my study fits into the context of the larger discussion?

system and network monitoring

by John Sellens

John Sellens is Associate Director, Technical Services, with GNAC in Toronto. He is also proud to be husband to one and father to two.

<jsellens@gnac.com>



As computers have gotten smaller and networks have gotten bigger, most of us have found ourselves worrying about more and more machines and network devices. In the old days, the typical installation of a small number of central servers, a larger number of ASCII terminals, and a few point-to-point serial or network links meant that a lot of system monitoring could be handled by periodic manual inspection or a few shell scripts, cron jobs, and mail messages.

These days, when there seems to be at least one server for each possible function; when everyone has a machine with what used to be thought of as major processing power on their desk; when networks are bigger, more complicated, and smarter; and when everything from modems to printers to soda machines is network-connected, local monitoring just isn't enough. Virtually every site needs some form of distributed or network-based monitoring mechanism, if only to have some ability to keep track of the worst problems.

In this article I'll discuss what monitoring is, along with where and why you might want to use it, typical components of a monitoring system, and some criteria against which to measure different monitoring systems and tools.

This is the first article in a planned series on system and network monitoring. Future articles will examine a variety of monitoring software packages, measure them against the evaluation criteria, and attempt to discuss the pros and cons of each and identify where the software might be most appropriately used. I'll primarily be examining open source and freely available software, but I'll also try to cover some commercial packages.

And, for the benefit of those who are unfamiliar with professional concert sound-reinforcement systems, I promise that I'll try to avoid attempting weak puns about asking for more SNMP in the monitors.

What Is Monitoring?

Monitoring is primarily intended to identify what has gone wrong or is about to go wrong. In general, monitoring systems can be thought of as having four components:

- Data collection and/or generation
- Data logging or storage
- Analysis, comparison, or evaluation
- Reporting and exception alerting

Basically, you collect some data points, stash them somewhere, compare them against established limits or failure indicators, and raise a flag if something's wrong.

That is a bit of a simplification, but the basic truth is there. Fortunately, most monitoring systems are a little more sophisticated than the bare-bones description above.

Data Collection

Data collection usually takes a few different forms, but most forms can be classified as some sort of probe. Some examples of common probes are:

- ICMP pings to indicate network connectivity
- Simple port probes (e.g., does a TCP/IP connection to port 80 succeed?)
- SNMP queries to determine specific states or activity levels

More sophisticated logging mechanisms can make it easier to identify trends or multiple failures that are due to a single cause.

(SNMP is, of course, the Simple Network Management Protocol – for an introduction to SNMP, see Elizabeth Zwicky's articles in recent issues of *;login:*.)

Data points can also be generated and submitted by a system or network element to the monitoring system, through SNMP traps, mail, or some other form of network connection. An obvious example of this is the use of a centralized syslog host that receives syslog messages from various hosts on the network.

Data Logging

In many cases, the collected data is logged in a fairly basic way, often through syslog or some flat file. Some systems log every data point they receive or generate; some log only the "interesting" ones.

More sophisticated logging mechanisms can make it easier to identify trends or multiple failures that are due to a single cause (such as a loss of connectivity that's due to a router or communications link failure). Logging mechanisms such as relational databases can add some complexity, but can also make certain kinds of reporting easier and more effective.

Analysis

In most cases, monitoring analysis takes the form of an immediate, realtime, good/no-good decision. For example, failed pings would normally be assumed to indicate a machine or network connection that's down, and a disk that reports 99% full may call for some attention.

Some systems can correlate multiple failures that are due to a single cause (that communications link mentioned above), and some can react or escalate after multiple consecutive failures (e.g., call your boss if the Web server doesn't respond for the third time).

The other type of analysis that is sometimes overlooked is trend reporting — if you can notice trends while they are happening, you'll have a better chance of adding more disk space to the /news partition before you run into major problems. I haven't (yet) seen this myself, but it would be nice, in a twisted sort of way, to get an automated message noting that disk use has been increasing, and that if current trends continue, the disk will be full in 42 hours.

In general, better analysis gets you better results, but you'll likely pay for it in increased complexity and increased cost.

Reporting

Probably the first type of reporting that people think of in relation to monitoring systems is alpha or numeric pager messages (which of course always come at the worst possible time). But there is far more to a proper reporting system.

Reporting is generally concerned with three types of information:

- Exceptions – problems that should be reported in some form of "alert" for investigation, action, and resolution.
- History – specific data for specific time periods, for such uses as traffic-level and outage reporting, as well as usage or capacity-based billing.
- Trends – aggregated (typically) data used for trend analysis and capacity planning.

In general, two styles of exception reporting are used with monitoring systems: report everything, or report only the problems. And, furthering that distinction, do you report events only as they happen, or do you report on the current state, identifying all "unresolved" issues?

How should exceptions be reported? A number of mechanisms can be used individually or in combination.

- One-time messages, pager, email, fax, and the like. These are most useful at the time of the first identification of a given problem – getting paged every few minutes about the very problem that you're working to solve can be somewhat distracting. But in the absence of 100% reliable communication, an effective method might keep sending alerts until they are acknowledged.
- Full-screen ASCII or Web-based status screen. These are often seen in "showplace" Network Operations Centers (NOCs) and list outstanding alerts in order of priority, or in forward or reverse chronological order.
- Query lists that can be reviewed periodically. This is perhaps most often seen in trouble-ticket systems.

The key being, regardless of reporting mechanism, that exceptions must be reported on a timely basis, and there should be some form of tracking mechanism so that unresolved problems are less likely to get lost or moved to the bottom of the to-do list.

Historical-data reporting is often built specifically for a particular intended purpose, with reports tailored to provide the needed information in the most effective manner possible. This is often used for client usage-based billings, or periodic outage or performance reports. Historical data is often needed only for a limited time, after which it can be archived or deleted.

Trend reporting is often best understood when visualized in some graphical format, making it easy to see where things are headed. The data used for trend reporting can often be aggregated as time passes. For example, you may want to have complete detailed data collected every five minutes for network bandwidth usage for the past week, but you may only want something like monthly average, maximum, and minimum peak usage for last year. This means that a lot less data needs to be stored and analyzed to generate trend reports.

You should use some form of system and network monitoring any time you have a system or network that someone cares about or relies on.

Where and Why Would You Use Monitoring?

Quite simply, you should use some form of system and network monitoring any time you have a system or network that someone cares about or relies on. Your particular circumstances will dictate what level of monitoring you will need and what style of monitoring system or systems will best meet your needs.

If you operate a nontrivial network, you'll probably want to monitor your routers, switches, and communication links for such things as bandwidth utilization, CPU and memory utilization, and interface-state changes. If you operate a collection of servers or a Web-hosting farm, you'll want to monitor things such as uptime, service availability, disk space and memory utilization, print queues, and network connectivity.

A properly functioning and configured monitoring system, with appropriate alert-dispatch mechanisms, is one more tool to help you identify and correct problems before your users and customers identify them for you.

Even in the absence of utilization-based billing and formal service-level agreements, historical and trend data can be very useful, providing evidence of the superior quality and availability of your systems and helping to provide additional data in support of your requests for additional equipment or personnel.

How Are Monitoring Systems Built?

Most monitoring systems are built from components, either real or virtual. Conceptually, most systems can be characterized as having the four components discussed above.

Monitoring systems come with some number of probes, either hard-coded into a larger program or implemented as separate programs communicating through some API or data-interchange format. There may be as few as two or three different probes or as many as 100 or more included with a particular monitoring system. Probes typically deal with a single type of query, sometimes restricted to dealing with equipment from a particular vendor.

There is often some form of "trap manager." An SNMP "trap" is a message sent by a device to alert a management or monitoring system of a change in state or an error condition, such as a failed component or a network interface going up or down.

Some form of configuration file or language is used to indicate which probes to direct at which devices, how often, and what the acceptable limits are for the values returned by those probes. There would also need to be some configuration information for use by the trap manager (if one exists), dictating what actions to take when traps are received, based on the type of trap, originating system, or other factors. And the configuration information would also need to indicate how to send alerts for the various types of exceptions as they are identified.

Beyond that, there is some form of logging, and some interface to tools for dispatching alerts (e.g., by mail, or calls to a system that sends messages to pagers), and various reporting tools. The reporting tools can range from rudimentary or nonexistent to sophisticated graphical interfaces with tools for filtering alerts prior to display, problem assignment, and so on.

In operation, a typical monitoring system becomes a series of interconnected "event loops," periodically probing, recording, dispatching, and resolving alerts, and logging data points for historical or trend reporting.

Evaluation Criteria

In future articles in this series I'll review various monitoring systems and software packages, and I'm going to use some or all of the following criteria as the basis for describing and evaluating those systems.

Size and Complexity

Monitoring systems range from very small (simpleminded ping tests that generate mail messages on failure) to very large (multiple probe machines, report generators, display and dispatch engines, and a high-availability database engine). Different organizations will need (and be able to cope with) different-sized monitoring systems.

Scalability

While closely tied to size and complexity, scalability is important if your network contains more than a small number of devices, or if you expect ongoing growth. A process or mechanism that works well against 20 or 30 machines can fail miserably in all sorts of interesting and painful ways when used with 200 or 2,000 machines.

Reliability

A monitoring system that is prone to failure, or that misreports failures or problems in other devices, can be worse than no monitoring at all.

Cost
Monitoring-system implementations can range from almost free (an hour's worth of work) to several hundreds of thousands of dollars (hardware, software, consulting, maintenance). My personal knee-jerk preference is for freely available software, but there are many instances where commercial monitoring products are the only reasonable way to go.

Number and Type of Probes
Can the monitoring system handle typical devices, more specifically, *your* devices? Can it be easily extended to handle other devices that only you have?

Configuration Complexity and Flexibility
A configuration format that can be machine-generated (from, say, an existing database), that can express a hierarchy of connectivity (so that you don't try to contact all the machines behind a gateway if the gateway is down), and has reasonable defaults and per-device customization is a very good thing to have. And if it's understandable and easy to maintain too, then so much the better.

Exception-Reporting Style
Does the monitoring system always display full status information for every device or service being monitored (which doesn't scale well), does it display only those devices that have unresolved exceptions, or does it offer the user a choice?

Exception-Reporting Tools
How are exceptions reported? Are there pager, email, fax, and Web interfaces? Is there a command line and curses-based text terminal interface and display? Is it easy to add additional exception reporting interfaces?

Logging and Data Storage
How is data logged – via syslog, flat files, a database? Are there mechanisms for data aggregation, de-duplication, and pruning?

Reporting Mechanisms
Are there suitable and flexible reporting mechanisms for historical and trend data? Is there a defined interface or tools to allow custom reporting?

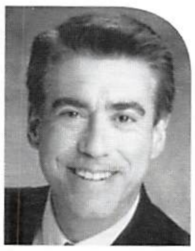
What Next?
That pretty much covers my introduction to system and network monitoring. In the next article in this series, I'll review one of the many popular and easily available monitoring packages. Let me know your favorite, and I'll attempt to explain how even though it may be the best one for you and it suits your 10base2-based network to a T-connector, it probably won't fit anyone else's situation (except that of the software author's, of course).

ten things to consider

by Paul Lemberg

Paul Lemberg is a business coach. To subscribe to Paul's free newsletter, send email to *<request@lemberg.com>* with the subject "subscribe".

<paul@lemberg.com>



[Editor's Note: Paul is a management consultant who occasionally distributes newsletters like this one. I found it especially interesting. – RK]

By now, you've set a working direction for the year, established clear-cut objectives. Your first-iteration plan to reach them should be in place. This now seems like an ideal time to rethink the whole thing, doesn't it? After all, one of the effects of Internet time is that plans are subject to change just as soon as – or perhaps even before – they are written. Along these lines of thinking, perhaps there are some items you missed. Maybe there are issues you didn't have time to consider, or even things your mind touched on but quickly passed over to deal with more urgent and pressing events. If you are off-cycle, and on the verge of a new period, you can use this exercise ex ante, rather than ex post. To help you stimulate your neural pathways and hopefully create an idea or two, I offer the following thoughts for your consideration. These "considerations" are not sequenced in order of importance. I think they are all important.

1. How far in the distance is your planning horizon? Most companies today plan 12–24 months out, calling anything beyond that "vision." Internet time implies a shortened time frame for activities, but does that time-collapse extend to a shortened vision as well? How much have you thought about what you will accomplish this decade? What will be your company's impact on the millennium? (OK – perhaps millennium is too far out. What about the century?) You may say you have more pressing fish to fry. Your investors would like to see increased returns sooner than that. While this might be true enough, taking the long view can inform the short view, leading to greater returns for years to come. What do you see when you take the long view?

2. How are your prospects' needs going to change? How is their world affected by the dramatic increases in connectivity and the compression of time? What are you doing to understand their changing environment – their changing business issues? What are you doing to improve your customers' business under these slippery conditions? To take it one step further, what do your customers' customers want? While you are at it, you might stop to consider, how are your suppliers' needs changing? Could those changes open up new opportunities for you, or darkly portend changes downstream totally derailing your business model? What about your distributors? Is their world shifting? Can you both benefit?

3. Who in your organization simply isn't contributing? As they say, your mileage may vary from individual to individual, but everyone has the responsibility to go some distance, to make something valuable happen. Not everyone will make good on that implied promise. The often-observed 80-20 rule applies to your staff as well: 20% of your people will produce 80% of the value. That leaves 80% producing only 20%. Do the math: the bottom 10% of your organization produce almost nothing. Who isn't making the cut? Should you be doing something about it? You may think it beneficent to provide that bottom percent with a paying job – don't. It isn't. The nonperformers know who they are, but they won't cut the cord on their own. Do what you can to help them reach the bar, but if after a while they don't make it, set them free to find an environment in which they can succeed. Free up your own resources for people who make a difference.

4. Are you creating solutions to today's problems? What about next week's, next year's, or the problems of several years from now? How are you figuring out what those problems are going to be, way out there on the time horizon? Because the solution you sell today should certainly address today's problems, but the solutions on today's drawing board better not. Who in your organization is responsible for trend-tracking and forecasting? Are you building scenarios for the future? What about prospect focus groups, or some other market-based feedback mechanism? Who is your resident futurist?

What do you believe about the business you are in?

5. What do you believe about the business you are in? For most people this is a strange question – we rarely spend time thinking about our own beliefs. The collection of beliefs you hold about your business – what the Germans call Weltanschauung – is decisive in most of the choices you make. How much risk to take. What's risky and what isn't. What projects and initiatives to undertake. What kind of resources you need and whom to hire. Whom to partner with, or should you have partners at all. Cooperate or compete. How to treat your team. What your customers should expect from you. How hard do you expect people to work? All these decisions stem from your beliefs, and it will help you to make them explicit. Once you surface those beliefs, you can start to distinguish which are useful beliefs and which are not. What is the benefit of a particular belief? Is this belief relevant to your current world, or is it a holdover from some past part of life? Then, when you are ready, you can experiment with new beliefs.

6. What are the obstacles to proceeding along your current path? Yes – you've set a plan in motion, and you are taking steps toward its achievement. But what roadblocks may rise up to stop you? What things could get in your way – foreseen and unforeseen? (I know, if it's unforeseen how are you going to see it? Use your imagination, that's the point of this exercise.) Rank these obstacles in terms of likelihood, then rank them in terms of severity. Consider how you might deal with them if they come up. The value of this is (a) like the Boy Scouts, you are better prepared; (b) you may illuminate issues you have been trying to sweep under the rug; and (c) you just may invent a whole new approach to get where you are going, and it just might be better than what you are doing now.

7. What, if you only knew how, would you be doing? What would you do now if you had additional resources – and should the lack of resources be stopping you? What, if you were sure it would be successful, would you jump on right away? What would you begin immediately, if your resources were limitless? (Yes, limitless can be relative.) What are you betting the future of your company on? What would you be willing to bet the future of your company on?

8. What are the most important issues, right now? Make separate lists for issues in your market and issues in your company. Which of these issues are you dealing with, which ones are on the back burner, and which ones aren't even in the kitchen? What are the processes you use to deal with these issues? Which issues are you ignoring, or hoping will go away? What breakthroughs might be possible by addressing or resolving issues in the latter category? Where are you "resolving" issues by compromising? What possibilities are available by refusing to compromise, or by breaking your compromises? What old stories or old ways of looking at things make these compromises seem inevitable? Where could new technologies (either material, virtual, or societal) be applied to break these compromises?

9. What are you sacrificing to accomplish your current objectives? The definition of sacrifice is giving up something of value for something of even greater value. Did you

What is the purpose of your organization?

intend to give up that thing of value, or is it a thoughtless byproduct of your other choices? Do not dismiss this lightly. In your business there are a number of priority-conflicting critical success factors. These include profitability, product development, new sales, customer satisfaction, recruiting and retention, revenue growth, sufficient capital – which one gets the most attention? And in this operating cycle – will each area get the attention it needs? Even in a lower position of priority, these areas cannot be neglected. What isn't getting done that needs to be done, and how are you going to do it?

10. What is the purpose of your organization? I don't just mean increasing shareholder wealth; that simply won't inspire your people to greatness. What besides that – a given – is the purpose of your company? Purpose is not something you invent, it is there already: you have to uncover it. Why do you come to work each day? What do you hope to accomplish in the long run? What about your executive team? Your individual employees – why do they come? What do they think they are doing each day? Do you know? Have you bothered to find out? You've just completed a planning cycle, and I'm asking what your purpose is! If you can't answer this question easily, now would be a great time to start.

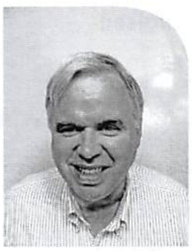
Bonus question for consideration: Are there any questions I've listed above that you do not have easy answers to, but wish you did?

Every so often I do an exercise called the "One Hundred Questions." If you would like a copy of my most recent 100 questions, along with how to use this simple thought-provoker, send an email to <*request@lemberg.com*> with the subject "100questions".

yep!

by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.



<scj@transmeta.com>

and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and trouble-shooter for technical companies.

<dustywhite@earthlink.net>

The human mind cannot directly remember a negative. Some ingenious research involving reaction times has demonstrated that when we remember a negative concept, we first remember the positive concept, and then associate with it a one-bit NOT tag. This is easy to demonstrate to yourself – if I say "Don't think of an elephant," what's the first thing you think of? An elephant!

So consider what happens when you express an intention or a goal as a negative, something you want to move "away from." Suppose, for example, that you want to earn money so you will not be in poverty. Every time you think "Not Poverty," your brain first brings up the concept "Poverty," complete with all its associations of filth, disease, powerlessness, hunger, ignorance, debt, and desperation. And while these associations are being lit up in your brain, there is one little bit saying "not! not! not!"

Is this the kind of stuff you want your brain reinforcing?

On the other hand, suppose you can visualize wealth, or, if that's too extreme, comfort. A warm room in a cozy house, furnished with taste, a good meal before you, bills all paid, and some savings in the bank. What lights up in your brain with this image?

Not only is this image more energizing to think about, it also puts you in a more receptive frame of mind in which you recognize and can act on pieces of your vision as they become possible.

When you become more aware of the power of positive thinking (to quote an old book title), you'll find it really amazing how many ways we reinforce our old negative patterns through "away-from" thinking.

At work, we raise our anxiety trying not to fail, rather than trying to succeed.

At the store, we worry so much about being cheated or making a bad purchase that the joy of ownership is blunted.

In relationships, we say, "I'm never going to do that again!" over and over. If we focus on, for example, our fears that our partner is unfaithful, think what circuits light up in our brain. Is it any wonder then that romance dies?

You can carry positivity one step further, not just thinking the thought, but walking the walk. You can act as if your positive desires are already obtained. You are a success at work, a savvy shopper, and your partner loves you beyond measure.

Dreamland? Not really. Realize that bad things do happen, and what you desire may not always come true. But by acting as if you have the positive things you desire, when good things happen you are in an ideal position to take advantage of them, and when bad things happen, you are also in an ideal position to find the greatest possible good in them.

So spend some time on your goals. Get rid of the "away froms." Make your goals vivid, see them, smell them, taste them, touch them. And then live as if you have them. One day you will realize that you do.

the hostile workplace

by John Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation, and other technology issues.

<John.Nicholson@ShawPittman.com>

In last issue's column, I discussed the concept of a "hostile workplace" and the need for companies to monitor the behavior of their employees. Having introduced the concept, I'd like to take this opportunity to discuss workplace harassment and explain something that frequently confuses people about freedom of speech (and other Constitutional rights). While this topic is not directly related to computers and technology, it is something that managers need to know and understand.[1]

Reader Questions

Before we get to the hostile workplace, however, a reader of the last issue's column noted that although I discussed the rights (or lack thereof) that an employee of a company has with regard to privacy of computer files stored on company computers and sent through the company network, I did not address any right to privacy or other rights that a third party (nonemployee) sender of an email message might have regarding how that message is treated.[2]

As I discussed in the April issue, Title 18 Section 2701(a) makes it a crime to access a system without authorization and to obtain, alter, or prevent authorized access to a wire or electronic communication while it is in electronic storage in such system,[3] except that "Subsection (a) of this section does not apply with respect to conduct authorized – (1) by the person or entity providing a wire or electronic communications service; (2) by a user of that service with respect to a communication of or intended for that user."[4] This language means that if you send an email message to a person, then any company whose network that message passes through can probably access and store that message, including, if you send the message to the person's work address, the recipient's employer. If the recipient then stores the message on a computer provided by the employer, then it would be just like the employee receiving a written letter and putting it in the company's files.

What Is Harassment?

Harassment is employment discrimination consisting of unwelcome verbal or physical conduct (such as comments, jokes, or acts) relating to the victim's constitutionally or statutorily protected classification (such as race, religion, gender, ethnic origin, or age) that has the effect of substantially interfering with a person's work performance or of creating a hostile work environment.[5]

According to the courts, speech in the workplace can be punished as workplace harassment if it:

- is severe or pervasive enough to create a "hostile work environment"
- is based on criteria including, but not limited to race, religion, sex, national origin,[6] age, disability (including obesity),[7] military membership or veteran status,[8] or, in some jurisdictions, dishonorable discharge from the military,[9] marital status,[10] family responsibilities,[11] sexual orientation,[12] personal appearance,[13] cross-dressing,[14] political affiliation,[15] criminal record,[16] citizenship status,[17] student status ("matriculation"),[18] receipt of public assistance, [19] or even smoking or use of tobacco outside the course of employment [20]

for the plaintiff and for a reasonable person.

Prior to the advent of email and the Internet, employers and employees did not have as much to worry about (although many of the "hostile workplace" cases come from the era before email and the Web). It was more difficult for speech or other behavior to be sufficiently "severe or pervasive" to create a hostile workplace. Employees had to actually tell each other jokes, either one at a time or in groups, or make copies of cartoons by hand. Employees could not email jokes, pictures, executables, links to Web pages, etc., around the company.

Now, however, the ease and speed with which information can be sent to multiple people (and sometimes the wrong people) creates a situation ripe for workers to be offended by their co-workers' sense of humor. Additionally, the casual and spontaneous nature of email may allow employees to write things that are disseminated beyond their intended audience and could be taken out of context. Moreover, the seeming privacy and anonymity of email and the Internet makes some people do or say things they would not do or say if they thought they might be seen or overheard by a third party. Unless employers can show that they have policies in place that prohibit such behavior and take action against those who violate such policies, employers can be held liable for substantial damages.

What Is "Freedom of Speech," and Does It Apply to Companies?

When a company places limits on what employees can say or wear or what posters they can put up, employees frequently claim that such rules are a violation of their right to free speech. Since the company is telling them what they can and cannot say, this seems to be true. At the same time, however, companies are being held liable for the behavior of their employees when the employees create a hostile workplace. This apparent conflict causes a great deal of confusion in the workplace. Frequently, neither the employees claiming the right to freedom of speech nor the person writing the corporate policy prohibiting harassment understands precisely what rights to "freedom of speech" are granted by the Constitution.

The First Amendment to the U.S. Constitution states, "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the Government for a redress of grievances."[21]

The key language in the First Amendment is the first five words – "Congress shall make no law." The thing that many people do not realize about the Constitution is that it only controls what the government can do. Thus, on private property, generally, as long as the restrictions are applied to all employees equally, a company can impose whatever regulations on speech or expression (putting up posters, etc.) it wants without violating an employee's constitutional rights.

What Should an Employer Do?

1. Develop a written harassment policy statement. This policy statement should begin by stating that harassment is illegal and will not be tolerated. The policy statement may further include the employees' right to work in an environment free from harassment and from retaliation for reporting harassment, the fact that harassment is a violation of state and federal law, identification of specific behaviors that constitute harassment (like those noted above), and an outline of consequences for engaging in harassing behavior.

2. Communicate the policy by posting it in the workplace and including the policy in employee handbooks or policy manuals.

3. Develop procedures that will be followed upon filing a claim of harassment and identify the person(s) to whom the employee should report the harassment.

NOTES

[1] This article provides general information and represents the author's views. It does not constitute legal advice and should not be used or taken as legal advice relating to any specific situation.

[2] For a discussion of an employee's right to electronic privacy in the workplace, see "Electronic Privacy in the Workplace," in the April 2000 issue of *;login:*.

[3] Section 2701(a) states: "Offense. Except as provided in subsection (c) of this section whoever (1) intentionally accesses without authorization a facility through which an electronic communication service is provided; or (2) intentionally exceeds an authorization to access that facility; and thereby obtains, alters, or prevents authorized access to a wire or electronic communication while it is in electronic storage in such system shall be punished as provided in subsection (b) of this section."

[4] 18 U.S.C. 2701(c).

[5] Merriam-Webster's Dictionary of Law (1996) *<http://dictionary.findlaw.com/scripts/results.pl?co=www&topic=7c/7cea1d560bd690325e45218463669979>*

[6] See, e.g., Harris v. Forklift Sys., Inc., 510 U.S. 17, 21-22 (1993) (barring harassment based on race, religion, sex, or national origin).

[7] Eggleston v. South Bend Community Sch. Corp., 858 F. Supp. 841, 847–48 (N.D. Ind. 1994) (barring harassment based on age and disability under the Age Discrimination in Employment Act and the Americans with Disabilities Act).

[8] 38 U.S.C. §4311 (1994) (barring discrimination against present or former armed service members). Additionally, several states, including California, Colorado, Florida, Illinois, Iowa, Michigan, North Carolina, Oklahoma, Pennsylvania, Rhode Island, Wisconsin, and Wyoming, have passed statutes that prohibit discrimination against present members of the armed services and/or the National Guard.

[9] Ill. Stat. Ch. 775 §§5/1-103(Q), 5/2-102 (1997) (barring discrimination in "terms, privileges or conditions of employment" based on "unfavorable discharge from military service").

[10] See, e.g., Cal. Gov't Code §12940(h)(1) (West 1992 & Supp. 1995) (barring discrimination based on marital status).

[11] D.C. Code Ann. §1-2512 (1981 & Supp. 1988) (barring discrimination in "terms, conditions, . . . or privileges of employment" based on "family responsibilities").

4. Finally, charge employees with the responsibility to report any harassment or other discriminatory practices.

You Hear So Many Ridiculous Stories . . .

Just like many of the stories of children being suspended from school for bringing aspirin or a squirt gun that looks too realistic, there are lots of stories about people objecting to things that seem harmless but that employers remove because of a complaint. For example, in one of the more extreme cases, a harassment complaint was filed against a graduate student who had on his desk a 5" x 7" photograph of his wife in a bikini. The employer ordered that the photo be removed.[22]

Unfortunately, because harassment law is potentially so broad (it applies to any conduct that is "severe" and "pervasive" enough to create a hostile workplace), because it operates based on aggregate effect rather than specific incident, and because the potential liability and publicity associated with a lawsuit can be so severe, companies must respond to each individual complaint. If a company were to ignore some complaints while responding to others, the company would effectively be saying that some conduct that is offensive to a particular employee is acceptable while some other conduct that is offensive to another employee is not acceptable. By doing this, the company could open itself up to liability.

The potential for workplace harassment creates a difficult environment for companies. On one hand, employers do not want to be draconian and punish workers for seemingly petty offenses. At the same time, however, any individual comment, jokes, or action could, when taken in the aggregate with all of the other comments, jokes, or actions, be the straw that breaks the camel's back for an individual employee. To avoid the risk of creating a "hostile workplace," an employer cannot simply tell all of its employees not to do or say so many offensive things that the sum of all of the offenses would create a hostile workplace. There is no way for any employee to know what other employees are doing or saying at all times. One employee may be present in different groups on different occasions when a single employee or even different employees make similar comments or tell similar jokes that are offensive to that one person. These separate events could be interpreted by a judge or a jury to be sufficiently "severe" and "pervasive" to create a hostile work environment. Because there is no way for any individual employee to know whether other employees are making similar jokes or comments or are doing or saying enough other things that the result of the collective actions is to create a hostile workplace, an employer has to prohibit all potentially offensive behavior and respond to each complaint equally.

Conclusion

Harassment is any speech or other behavior that, if "severe" and "pervasive" enough, can create a hostile workplace. The First Amendment protections for freedom of speech generally do not apply in the workplace; they apply only to government action. Because the terms "severe" and "pervasive" are so vague, to protect themselves from liability, employers must establish and enforce policies that restrict any speech or activity that, if repeated enough times or by enough people, might be held by a jury or judge to be "severe" or "pervasive" enough to create a hostile workplace.

[12] Leibert v. Transworld Sys., Inc., 39 Cal. Rptr. 2d 65, 67 (Ct. App. 1995) (barring harassment based on sexual orientation).

[13] D.C. Code Ann. §1-2512 (1981 & Supp. 1988) (barring discrimination in "terms, conditions, . . . or privileges of employment" based on "personal appearance").

[14] New Orleans Code §86-1 (stating that discrimination based on "gender identification," which includes cross-dressing, is to be treated as discrimination based on sexual orientation), 86-131 (barring discrimination based on sexual orientation, defined to include discrimination "with respect to . . . terms, conditions or privileges of employment," which includes hostile environment harassment).

[15] D.C. Code Ann. §1-2512 (1981 & Supp. 1988) (barring discrimination in "terms, conditions, . . . or privileges of employment" based on "political affiliation").

[16] N.Y. Correction Law §752 (generally banning discrimination based on having "previously been convicted of one or more criminal offenses").

[17] Ill. Stat. Ch. 775 §5/2-102 (1997) (barring discrimination in "terms, conditions or privileges of employment" based on "citizenship status").

[18] D.C. Code Ann. §1-2512 (1981 & Supp. 1988) (barring discrimination in "terms, conditions, . . . or privileges of employment" based on "matriculation").

[19] Minn. Stat. Ann. §363.03(2) (barring discrimination in "terms, conditions, . . . or privileges of employment" based on "status with regard to public assistance").

[20] D.C. Code Ann. §1-2512 (1981 & Supp. 1988) (barring discrimination in "terms, conditions, . . . or privileges of employment" based on "smoking or using tobacco or tobacco products outside the course of . . . employment").

[21] U.S. CONST. amend. I.

[22] Nat Hentoff, "A 'Pinup' of His Wife," Washington Post, June 5, 1993, at A21. The law's ban on sexually suggestive materials in the workplace is not limited to those containing nudity; see, e.g., In re Butler, 166 Vt. 423, 697 A.2d 659, 664 (1997) (concluding that "a poster of a woman in a skimpy bikini" could count as harassment, because "the posting or display of any sexually oriented materials in common areas that tend to denigrate or depict women as sexual objects may serve as evidence of a hostile environment").

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He has just become Chief Knowledge Officer (= Mr. Know-It-All) at Matrix.Net.

<peter@pedant.com>

What If It Ain't English?

While a large number of people in the world speak and read a version of English (as a first, second, or *n*th language), only a very few cultures employ the 26 characters of our alphabet. Representing the various scripts becomes vital. Just how many we should be trying to accommodate is totally unclear.

In 1996, Oxford University Press published *The World's Writing Systems*, a long-awaited tome edited by Peter T. Daniels and William Bright. It's over 900 pages and costs about $150. I consider it a treasure.

When *Unicode Standard*, Version 1.0 appeared in 1991 (vol. 1) and 1992 (vol. 2), I was pleased, but not overwhelmed: there were a number of gaps, and a large number of unattractive choices had been made.

2.0 (1996) was a vast improvement. 3.0, the 1,000-page, large format at hand, is yet better. But its choices are quite strange at times.

For example, under "European Alphabetic Scripts," there are descriptions for Runic and for Ogham. I presume that some elderly Scandinavians and old Irish use these in sufficient quantity. (The Unicoders don't seem to realize that there were non-Scandinavian runes.) On the other hand, I gather that the 25 million folks in Afghanistan carry too little weight for the Pashto version of Arabic to be treated. Under Cyrillic, Byelorussian and Ukranian are treated as variants of Russian letter forms; Glagolithic is unsupported.

South Asia has its problems, too. It might have been helpful for the Unicoders to refer to Daniels and Bright. In Unicode 3.0 (p. 211), it claims that Santali is written in "Devanagari." In Daniels and Bright, Norman Zide tells us that Santali is written in Ol' Cemet as well as "four 'older' scripts: Devanagari, Bengali, Oriya, and Roman." (Other "tribal" groups have also adopted Ol' Cemet.)

I won't beat the Unicoders more than this. There are hundreds of millions of people on this planet whose languages/scripts are not well done-by by the Unicoders. But I recognize that this is a work-in-progress, and that we won't require Pashto or Santali speakers to employ English when they get PCs.

For those who do scholarly work, I suggest getting a Mac and the font diskettes/CDs from Lloyd Anderson at Ecological Linguistics or the Egyptian hieroglyphic fonts from Cleo Huggins (based on the Gardiner OUP fonts).

I trust that someday even Akkadian, Babylonian, Hittite and Sumerian cuneiform, and Mayan hieroglyphs will find their way into Unicode.

But, after nearly a decade, we ain't quite there yet.

BOOKS REVIEWED IN THIS COLUMN

THE UNICODE STANDARD, VERSION 3.0
THE UNICODE CONSORTIUM
Reading, MA: Addison-Wesley, 2000.
Pp. 1,040 + CD-ROM. ISBN 0-201-61633-5.

[INCR TCL/TK] FROM THE GROUND UP
CHAD SMITH
Berkeley, CA: Osborne/McGraw Hill, 2000.
Pp. 746. ISBN 0-07-212106-8.

PRINCIPLES OF NETWORK AND SYSTEM ADMINISTRATION
MARK BURGESS
Sussex, U.K: Wiley U.K., 2000. Pp. 400.
ISBN 0-471-82303-1.

THE PROCESS OF NETWORK SECURITY
THOMAS A. WADLOW
Reading, MA: Addison-Wesley, 2000. Pp. 283.
ISBN 0-201-43317-6.

Tcl Me

The successor to C wasn't either P or D, but C++. Following this example, Tcl led to Michael McLennan's "extension," [incr Tcl/Tk]. [incr Tcl/Tk] is useful for developing large-scale applications and building extensible GUIs. Chad Smith has an engaging style and has really produced a volume that will teach and inform: you need not be a Tcl programmer to understand this book. As Tcl isn't O-O, if you want to employ O-O capabilities, you're going to need [incr Tcl/Tk]. And if you aren't an adept, you'll want Smith. This is a fine book.

Another Winner!

I keep Nemeth et al. and Aeleen Frisch at hand for referencing the system admin tasks we all need to do. Burgess's fine book is something "completely different." It is a well-articulated introduction to a corpus of guiding principles for system administrators.

And as we live in a world of heterogeneity, Burgess "covers" UNIX, UNIX-like, DOS, Windows, Mac, Amiga, and NT systems.

Burgess says that he wants to express a sound and logical way to approach networked systems. While I can find nits (that's a reviewer's job, isn't it?), I consider this an important book. More and more talk of certification can only lead to a body of knowledge and a set of tenets that are "required."

I think that Burgess will become part of the required reading for future (and current) system administrators.

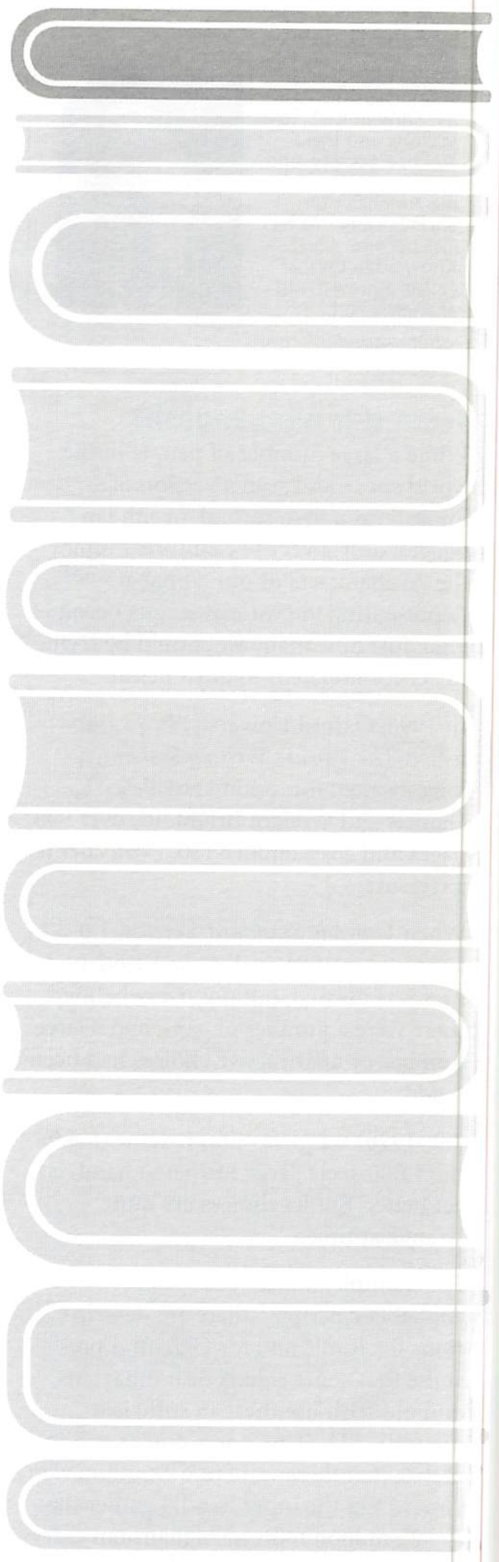
Are You Secure?

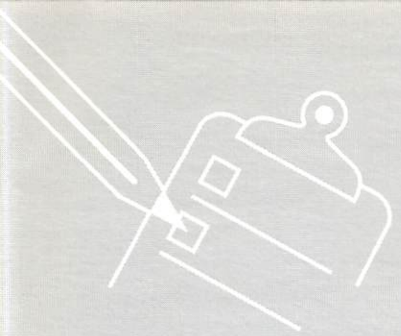
I feel that I've been reading and writing about security for too long a time. But Tom Wadlow has turned out a fine, small (under 300 pages) volume for the network-security administrator. As Wadlow points out, "If you are a five-person real estate office, with individual Internet dialup connections for all your Windows 98 desktop PCs, this is not the book for you." The only real criticism I have of this book is that Wadlow gives no "further readings" nor a bibliography. I wish that Bill Cheswick, Dorothy Denning, Gene Spafford, Simson Garfinkel, and many others were placed appropriately. Even the citations from Sherlock Holmes are merely ascribed to Conan Doyle.

Collections

If you go to the IETF site, look up RFCs, and then download and print them, as I do, you end up with oceans of paper, much of which flutters like leaves in the wind, shedding all over the keyboard, desk, floor, etc. I also have about six feet of files. Morgan Kaufmann (a division of Harcourt Brace) is performing a good deed. They have gotten Pete Loshin to edit a series containing the full RFCs by subject.

The first volume was *The Big Book of IPsec RFCs*, the second was *The Big Book of World Wide Web RFCs*, and the third was *The Big Book of Internet File Transfer RFCs*. I have found them quite useful but cannot review them: I wrote the prefaces to the ftp volume and to the (forthcoming) *Big Book of IPv6 Addressing RFCs* – and have other involvement as well. But they are worth looking at, if only as a way to stop deforestation.





Standards Reports

Our Standard Report Editor, David Blackwood, welcomes dialogue between this column and you, the readers. Please send your comments to *<dave@usenix.org>*.

News About Standardization

by Keld Simonsen

Keld Simonsen is active in ISO standardization, particularly in internationalization, POSIX, C, C++, and making it all available on the Web.

<keld@dkuug.dk>

This is my first time reporting to *;login:* readers about recent and forthcoming events in the standards world. I am doing this out of Europe, and primarily via the ISO channels, so my angle will be a quite international one. My interests in standardization are mostly in internationalization, UNIX/Linux, and the Net. Being non-American and not a native speaker of English is quite a disadvantage in the standardization world, and the difficulties make me work hard for true international, open cooperation on standards, especially within ISO and the Internet Society.

This time I will report briefly on news in internationalization, Java, C, C++, and UNIX/POSIX.

INTERNATIONALIZATION

A new standard on ordering of strings, covering all of ISO 10646, is about to be approved as ISO/IEC 14651. The final ballot is likely to be issued in April, with approval in June. The tables of the standard are planned to be available from the ISO server at *<http://www.iso.ch/>* – and there will most likely be a pointer to it from the Web of WG20 at *<http://www.dkuug.dk/jtc1/sc22/wg20/>*.

Another standard, or actually technical report, is hopefully also to be finalized. This is on how to specify cultural conventions, in an enhanced POSIX locale-compatible style. This TR has support for ISO 10646, the big 32-bit character set, and the above-mentioned sorting standard, and it has been implemented for glibc 2.2. The Americans in the WG20 working group, who are quite Unicode-oriented, are not amused with this specification, and they were some of the prime movers for making it a technical report, because it does not build on Unicode file formats and Unicode data, but rather on POSIX-compatible formats and data that have been widely employed in Linux and available from the ISO POSIX working group.

A standard on internationalization APIs, ISO 15435, is in the works. It covers the functionality needed for the data formats defined in the new cultural-conventions technical report. There is also some American resistance in the WG20 working group, as many companies have already made their own specifications for the functionality of these APIs. My take is that we really need a standard to provide portability in this area to facilitate support for the very important area of internationalization in as many applications as possible, and it should be as consistent as possible. The APIs build on C and C++ functionality but have been enhanced to cater to threads and improved functionality as described in the new locale standard mentioned above. Some interest in implementing this standard has been expressed in the Linux communities.

A third standard in the battle zone is the revision of the standard on the cultural register (ISO 15897). The Americans are once again not happy with such a standard, which is made in Europe and outside their control, and which builds extensively on the POSIX standards. In this case, the standard is already fully approved and applicable today. The data registered with the standard, including POSIX locales and char maps, is available from *<http://www.dkuug.dk/cultreg/>*. A number of countries in Europe are in the process of delivering their locales to the register.

Common to these standards is that they all build on C and POSIX technology, they are developed in the open standards world of ISO, and they are not under the control of the Americans. The Americans gathered in the Unicode consortium try to pull their standards through, which is quite understandable. The problem is only that these are standards on cultural issues, where each country should know best how their culture behaves and what serves them best. And here the Silicon Valley–based Unicode consortium tries to decide the culture for all countries of the world.

This work is also connected to the discussion about whether an association should be involved in writing the standards, or be giving out information on standards. I think that we should do both. In some cases, if we did not participate, there would not be anything to tell about. For example, there would have been no internationalization support in POSIX without the pressure from DKUUG, EurOpen, and USENIX, and 8-bit ESMTP would not have happened. The above-mentioned standards on the cultural register, the new locale standard and the i18n API standard, rely heavily on work from associations. It is also important that it should be a democratic process that determines how vital parts of our common information-age future should look. It should not be just a single or a couple of dozen large firms that decide how most computers in the world should look. So, participate!

More information on these internationalization standards, including the latest drafts, is freely available from the WG20 Web site at <*http://www.dkuug.dk/jtc1/sc22/wg20/*>.

NEW UNIX/POSIX STANDARD

The cooperation among IEEE, The Open Group (X/Open), and ISO is now underway for the common UNIX/POSIX standard. Everybody can participate in the process; look at <*http://www.dkuug.dk/jtc1/sc22/wg15*> for the Austin Group. It is a revision of the whole suite of UNIX and POSIX standards. The plan is to make just one document, based on the UNIX 98 Single UNIX Specification, and the same document will serve as the standard in all of the three participating organizations. It is not clear to me, though, whether the name on the standard will be UNIX or POSIX. It is expected that the Austin Group will send the combined document for ballot in May 2000 to the three participating organizations. I look forward to seeing how this three-way balloting procedure will work.

JAVA STANDARD WITHOUT SUN?

Sun has withdrawn its application to make Java an ECMA standard. This is because Sun wants full control over the standard, and according to the procedures of ECMA, ECMA will themselves have full control over the standard (that is, the firms cooperating in ECMA, e.g., IBM, HP, Microsoft). Sun tried earlier to get Java accepted as an ISO standard; however, Sun withdrew that application too, because ISO also demanded full control over the process.

ECMA and ISO are now considering producing Java standards without Sun. These should in the first instance be the Java Virtual Machine and Java Core Language. It is thought to be feasible to make a standard without Sun's cooperation, because there is much text that can be recycled from the C and C++ standards and from textbooks on Java. There are previous instances of standardization without the cooperation of the originators, e.g. POSIX!

Participation in the ISO Java Study Group is open to all free of charge; send mail to <*sc22jsg-request@dkuug.dk*> with a body containing the word "subscribe".

C++ DOCUMENTS ACCESSIBLE

If you are interested in C++, then you will be pleased to hear that all the technical papers from the WG21 working group are now freely accessible at <*http://www.dkuug.dk/jtc1/sc22/wg21*>; look for "papers." ISO has agreed to a new plan to make a technical report about support on embedded processors, with a proposal about portable support for hardware I/O. WG21 is also processing defect reports on the ISO C++ standard from 1998, and the corrections approved to the standard are freely accessible at the above URL (see issues lists).

C STANDARD APPROVED

The new ISO C standard C99 was published in December 1999. See <*http://www.dkuug.dk/jtc1/sc22/wg14/*>. WG14 is currently working on a rationale for the new standard, with explanations of how the new components should be used. This forthcoming rationale will become freely available at the above URL.

FAQ: ISO/IEC 10646-1 Versus Unicode

by Alain LaBonté

Alain LaBonté is the head of the Canadian delegation to ISO/IEC JTC1/SC22/WG20, the internationalization (i18n) working group.

*<alb@sct.gouv.qc.ca>

Q. Is the ISO/IEC 10646-1 Standard the same thing as Unicode?

A. It is pretty close, yes. The Unicode standard is more restrictive.

Q. Then how is the ISO/IEC 10646-1 Standard different from Unicode?

A. The ISO/IEC 10646-1 Standard is an International Standard (with a capital S – only the ISO can claim to have it) that covers:

- 16-bit or 32-bit code;
- "transformed formats" for compatibility with existing transmission standards;
- three levels of compliance for the internal representation of characters:

1. no composition of characters (all characters fully formed instead of including basic characters followed by diacritics) – this excludes many languages but simplifies the life of programming languages for all Western languages without exception (and for languages that do not require composition, such as all Far Eastern languages);
2. the composition of some, but not all, characters (obscure level, not sufficiently thought out; will be little used, in my opinion);
3. the mix of the technique of composition with the possibility of coding fully formed characters;

- total openness in the use of characters (no canonical form, no equivalency between composed characters and fully formed characters);
- possible support for dead languages, in addition to all the living languages;
- developmental possibilities for all practical purposes unlimited (eventually up to two billion separate characters).

Unicode provides:

- exclusively 16-bit code;
- a transformed format to allow access to at most a million 32-bit coding characters from ISO/IEC 10646-1 (this is considered amply sufficient for the foreseeable future, even long term, for business purposes);
- a canonical form allowing for "normative" equivalency of characters that are precomposed or formed in a predetermined order from a base character and diacriticals;
- rigid methods of presentation (no exceptions);
- in parallel with this, various other closed methods of processing and presentation (the advantage is that implementation is rigorously predictable); what is noteworthy is that this standard is directly related to a classification method that constitutes a "delta" framed within the ISO/IEC 10651 International Standard.

These are the essential differences, but the coding is essentially the same. Whatever complies with Unicode complies with the International Standard, but the opposite is not necessarily true.

Q. Is Unicode a standard?

A. In the strict sense of the word, it is a de facto standard, and therefore a "private" norm. During a recent debate, the conclusion was that standardization was a process that also included the development of de facto standards. A "de facto standard" is therefore in some ways an object of standardization, but a "de jure standard" is much more than that; it is more general in scope. In short, the difference is relatively fuzzy. But usage tends to distinguish "de facto standards" from proper "de jure standards." (French distinguishes those two terms in single words: "de facto standards" are known as *standards* and "de jure standards" are known as *normes* in that language.) With respect to "International Standards," the sole authority that has the right to use this term is the ISO, the International Organization for Standardization, and this by agreement with the major international organizations (such as the WTO, the UN, ITU, etc.). This has importance because when you talk about an International Standard at the WTO within the framework of world trade, you are not referring to just any standard. In this sense, ISO/IEC 10646-1 legitimates the Unicode standard. This is the case for many other important standards.

Q. Sometimes I hear people talking in French about the "*Unicode standard.*" What is a "*standard*"? In my view, "standard" is the English word that corresponds to the French *norme,* but I have my doubts. People often talk about "*standard*" with respect to .doc files, but there is nothing more illogical or incomprehensible than that type of thing! Sometimes I have the impression that *standard* in French means something like: "This year your system can read it, but next year it will not be able to, so you will have to buy the new 'standard' version of the software!"

A. I could not have said it better. A de facto standard is like strands of spaghetti thrown against a wall. For a while they will stick, and you can count on them being there. If they do not stick, you forget about them. So a standard depends on what is going on at the time, on marketing, and not on a planned desire for consensus. The standard will create a drawing on the wall at some time, a Riopelle, consolidating everything into a consensus, turning the drawing into plastic.

An International Standard is basically a planned Riopelle, i.e., all the strands of spaghetti will not suddenly be thrown against the wall; they will be placed there carefully, one by one, and then the decision will be made as to whether it is an International Standard when 75% of the participating countries in a project are in agreement. People will even hasten to satisfy those who are particularly capricious and have not yet come on board, prior to publication, to the fullest extent possible. The bill for an International Standard is more impartial, but it is also a much more difficult path.

Q. What have UTF-7 and UTF-8 to do with all of this? How do they differ? Or are these two "standards" in the meaning described in the previous paragraph, that is, things that you need to avoid like the plague?

A. UTF-8 is a "transformed format" that has been standardized in ISO/IEC 10646-1, which breaks down each piece of complete 32-bit code into 8-bit pieces, each of which is able to pass through the most capricious 8-bit transmission devices (some 8-bit combinations may not be able to get through, specifically combinations representing control characters). UTF-8 eliminates this limitation by reducing the combinations of bytes used.

UTF-16 is a "transformed format" that has also been standardized in ISO/IEC 10646-1, which allows access to just sixteen 32-bit code planes (1 million characters, instead of 2 billion). Essentially created to break the Unicode standard out of its impasse (initially the Unicode standard was limited to 64,000 characters, but it quickly became apparent that that was not enough for all the Han characters, for example, which you may find particularly interesting); it was able to do this, but is not able to get around the barrier of the finicky transmission devices described above.

UTF-7 is a nonstandardized "transformed format" (which is allowed on the Internet) that can get past the archaic barrier of devices that are on their way out which allow nothing but the safe passage of 7 bits per byte (a waste of 16% of possible data on an 8-bit byte). I see a very limited future here.

UTF-32 is a *false* "transformed format," nonstandardized, defined by the Unicode consortium, which limits the 32-bit coding to the list allowed by UTF-16 and therefore by the Unicode standard. This is a guarantee that the 32-bit coding used is compatible with Unicode.

UTF-8 is the current trend, the best compromise, for all present and future communication protocols for an "internationalized" and "localizable" Internet. This format allows the unrestricted coding of the entire present and future inventory of the universal character set (ISO/IEC 10646-1) and, therefore, of Unicode. Processing it is, however, less simple than is processing the native 32-bit code. This is an intermediate format that will be especially useful in transmission.

The POSIX Bookshelf

by David Blackwood
Standards Reports Editor
<dave@usenix.org>

This is the second in a series of articles that will provide you with a list of published sources of information about the POSIX family of standards. This month I will list some of the titles available from commercial publishers.

Go Solo 2: The Authorized Guide to Version 2 of the Single UNIX® Specification, edited by Andrew Josey, The Open Group, May 1997, ISBN 0-13-575689-8.

Programming with POSIX Threads, by David R. Butenhof, Addison-Wesley, May 1997, ISBN 0-20163-392-2.

The POSIX Programming Environment, by S. Walli, Addison-Wesley, January 1997, ISBN 0-20163-345-0.

Pthreads Programming: A POSIX Standard for Better Multiprocessing, by Bradford Nichols, Dick Buttlar, and Jacqueline Proulx Farrell, O'Reilly & Associates, September 1996, ISBN 1-56592-115-1.

POSIX.4: Programming for the Real World, by Bill O. Gallmeister, O'Reilly & Associates, January 1995, ISBN 1-56592-074-0.

Open Systems Handbook: A Guide to Building Open Systems, by James Isaak, Kevin Lewis, Kate Thompson, and Richard Straub, The IEEE Standards Press, 1994, ISBN 1-55937-435-7.

UNIX, POSIX and Open Systems: The Open Standards Puzzle, by John S. Quarterman, Addison-Wesley, January 1993, ISBN 0-20152-772-3.

POSIX Programmer's Guide: Writing Portable UNIX Programs with the POSIX.1 Standard, by Donald A. Lewine, O'Reilly & Associates, April 1991, ISBN 0-93717-573-0.

The POSIX.1 Standard: A Programmer's Guide, by Fred Zlotnick, Addison-Wesley, April 1991, ISBN 0-80539-605-5.

O'Reilly & Associates also has a Web-based course, "Introduction to POSIX Threads." Information on this course is available at *<http://training.oreilly.com/courses.html#Posix>*.

If you have information about other POSIX-specific books, training, or other resources, please email me at *<dave@usenix.org>*. Thanks.

IEEE-SA Announcements

The IEEE Standards Association has recently announced that the Project Authorization Request (PAR) for P2003.1b (C/PA) Standard for Information Technology – Test Methods Specifications for Measuring Conformance to POSIX – Part 1: System Application Program Interface (API) – Amendment 1: Realtime Extension (C Language) has been revised.

It also announced that Standard 2003.1-1992 (C/PA) IEEE Standard for Information Technology – Test Methods for Measuring Conformance to POSIX – Part 1: System Interfaces has been reaffirmed and that P2003.1b/D6 (C/PA) Standard for Information Technology – Test Methods Specifications for Measuring Conformance to POSIX – Part 1: System Application Program Interface (API) – Amendment 1: Realtime Extension [C Language] has been approved.

These announcements and other information on the IEEE Standards Association can be found on their Web site at *<http://standards.ieee.org/>*. Complete details on the status of all POSIX projects are available at *<http://www.pasc.org/standing/sd11.html>*.

USENIX news

Election Results

The results of the elections for Board of Directors of the USENIX Association for the 2000–2002 term are as follows:

President:		
Daniel Geer	1,263	
		96 abstentions
Vice-President:		
Andrew Hume	1,262	
		94 abstentions
Secretary:		
Michael B. Jones	1,139	
		208 abstentions
Treasurer:		
Peter Honeyman	1,243	
		109 abstentions

Directors:

Jon "maddog" Hall	915
John Gilmore	892
Avi Rubin	853
Marshall Kirk McKusick	792

Not elected for Director:
Marcus Ranum
Dirk Hohndel
Darrell Long

Total number of ballots cast: 1,385
Total number of invalid ballots: 1

Newly elected directors will take office at the conclusion of the next regularly scheduled Board meeting, which will be held June 20, 2000, in San Diego, California.

Ellie Young
Executive Director, USENIX Association

So long, and thanks for all the fish

by Andrew Hume
President, USENIX Association
<andrew@usenix.org>

By about the time you read this, I will have finished my thralldom as President of USENIX. After four years as President, I will become Vice President, supporting our new President, Dan Geer.

I have quite enjoyed my role as President. In many ways, watching the organisation evolve and grow is similiar to watching my twins grow: astonishment at how fast they grow, pride at their achievements, and most of all, finding out how much fun they are. You, too, should be proud, and here are a few examples why:

- Membership continues to grow at an impressive 15% annually (our sister societies, such as ACM and IEEE/CS, have essentially static membership levels).
- We are renowned for the quality and diversity of our conferences and workshops, and the exemplary work of our staff, in particular Judy DesHarnais, in running these events.
- We have a large and effective program for funding undergraduate and postgraduate research programs.
- We are achieving recognition as a major technical society; at a recent Computing Research Association meet-

USENIX MEMBER BENEFITS
As a member of the USENIX Association, you receive the following benefits:

FREE SUBSCRIPTION TO *;login:*, the Association's magazine, published eight times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on security, Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

ACCESS TO *;login:* online from October 1997 to last month *<www.usenix.org/publications/login/login.html>*

ACCESS TO PAPERS from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web *<www.usenix.org/publications/library/index.html>*.

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, election of its directors and officers.

OPTIONAL MEMBERSHIP in SAGE, the System Administrators Guild.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMS from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. See *<http://www.usenix.org/membership/specialdisc.html>* for details.

FOR MORE INFORMATION REGARDING MEMBERSHIP OR BENEFITS PLEASE SEE
<http://www.usenix.org/membership/membership.html>
OR CONTACT
<office@usenix.org>
Phone: 510 528 8649

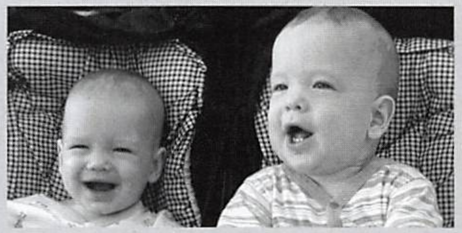
ing, I kept hearing phrases like "when associations like ACM, USENIX, and IEEE/CS."

The most important ingredient in all of this is, of course, you, the members. You keep paying your dues, coming to conferences, and sharing your experiences. In particular, I salute those of you who volunteer for running for the Board, being Program Chairs, and just helping out at conferences.

A close second, though, are the USENIX staff. Make no mistake, you elect officers and directors and they guide the organisation. But the staff runs it, and in USENIX's case, runs it extremely well. I, and the rest of the Board, rely heavily on Ellie Young, our Executive Director; I cannot overstate how well she does her job. Ellie is ably supported by Gale Berkowitz (Deputy Executive Director), Jane-Ellen Long (Production/IS Director) and Monica Ortiz (Marketing Manager).

And the whole organisation depends on Judy DesHarnais for her superb work in planning and running our various events. The rest of the staff are Bleu Castañeda, Moun Chau, Peter Collinson, Vanessa Fonseca, Dana Geffner, Dan Klein, Jim Lawson, Jennifer Radtke, Becca Sibrack, and Toni Veglia.

A final thanks goes to my family. My work with USENIX has involved a lot of travel; this used to be just inconvenient, but nowadays it means my wife has to look after Nicole and Jason, our 15-month-old twins, on her own. Many of you will understand what a load that is. So thank you very much, Karen.


The twins, Nicole and Jason

Karen

20 Years Ago in UNIX

by Peter H. Salus
USENIX Historian
<peter@pedant.com>

Last column I quoted Ted Dolotta on System IV and on the 7"x9" manuals. Here are a few more notes from Ted to keep you all amused.

When INTERACTIVE Systems was porting UNIX to the PC AT (PC/IX) and to VM/360 (VM/IX) under contract with IBM, we ran into IBM's "standard practices" folks.

They started to review the UNIX System III documentation and went ballistic – you know, parent processes that kill and/or abort their children, master-slave relationships, etc., etc.

They also objected to touch as . . . you guessed it: too intimate. They wanted to replace every occurrence of "touch" with "press." One day we were sitting discussing this weighty topic with IBM, and one of our secretaries, Joyce Yoshihata, overheard the discussion and said – to one in particular – "Gee, I did not know that I have been a presstypist for over 25 years." IBM capitulated on the spot.

They were more adamant about other words: demons, aborts, etc. I finally threw down the challenge: "You want UNIX, or you want to be the laughing stock of the UNIX user community?" They gave up, and I even kept them from censoring the BUGS section of one troff macro package manual page (which I had authored at Bell Labs) that said something like, "If you do thus and such, there will be bird-dropping-like things on your typeset output."

There were two subsequent developments:

One of my documentation guys created a manual page, doctor(6) – Section 6 of the manual being the Games section – whose sole purpose was to use all the words that IBM objected to, and to use them in the most offensive, non-PC (no pun intended) way possible. It's a masterpiece of offensiveness, but hysterically funny. I still have it on my system.

When we started doing AIX for the PC RT, IBM took over the documentation task: they sorted all the commands into the EBCDIC collating sequence, alphabetized all the flags on each manual entry (so some secondary flags were described before the primary ones which they modified, and the manual entries got *very* hard to follow), removed all the BUGS sections (IBM-sold code has no bugs, right?), and replaced all the words they did not like, including "touch."

And because the PC RT was a workstation, not a terminal, they did a global replace of the word "terminal" by "workstation," ending up, on the yacc(1) manual page, with sentences such as "The parser stops as soon as it encounters a workstation symbol," and other such gems.

One thing that I was rather pleased with is that we created the entire documentation set for VM/IX by taking the PC/IX documentation and globally replacing PC/IX with VM/IX, and then modifying a dozen manual pages and a few paragraphs in the administration guide to account for the fact that VM/IX was a multi-user system while PC/IX was single-user. But 99%+ of the documentation was identical. And it had all those "nasty" words in it, like real UNIX . . .

But I did – on my own – make one deletion: I read the entire list of fortune cookies – fortune(6) – and deleted the following fortune: "Question: How is Thomas J. Watson buried? Answer: 9-edge in, face down." I doubt that anyone who has not grown up with punched cards will even understand this, but it was funny – if somewhat offensive – in those days.

Thanks, Ted.

Note: In 1985 the USENIX Association held its 10th Anniversary blast in Portland, OR, with a salmon roast sponsored by Tektronix. It is not at all clear to me how this number was arrived at. If we're memorializing the first meeting of the UNIX Users' Group, that was May 1974. If it's the meeting at which the name was changed to USENIX, that was July 1977. If it's the organizational meeting to consider the bylaws for the new Association, that was November 1978. A few years ago, Evi Nemeth suggested to me that 1974 was year 0 and thus 1995 was 20. It works for me.

Notice of Annual Meeting

The USENIX Association's Annual Meeting with the membership and the Board of Directors will be held during the 2000 USENIX Annual Technical Conference in San Diego. The meeting will be held on Thursday, June 22, from 4 to 5:30 PM, in a room to be announced on-site.

Everyone is welcome!

MEMBERSHIP
Telephone: 510 528 8649
Email: <*office@usenix.org*>

PUBLICATIONS/WEB SITE
<*http://www.usenix.org*>
Jane-Ellen Long
Telephone: 510 528 8649
Email: <*jel@usenix.org*>

Perspectives on the SAGE Salary Survey

by Peg Schafer
SAGE Treasurer
<peg@usenix.org>

As I sit down to write this article on the 1999 SAGE System Administrator Salary Profile, Judge Jackson is sticking it to Microsoft, the stock market (i.e., my retirement fund) is bouncing like a ball, and another student of mine has just accepted a job offer that has a better compensation package than my own. Rarely does such daily news affect my personal life to such a degree!

The motivation of the SAGE Salary Profile was to give form to what we already knew: systems managers are in short supply and getting decent pay. But how much do they make? Are there differences among industries? Prospective systems managers would like to know if experience is worth more, in the long run, than an advanced degree. Beyond the obvious we knew very little about compensation for system administrators. As a result of this year's SAGE Salary Profile, we now know more, but, as with most research, new questions arise. Working with the Human Resources Research Organization (HumRRO), we now have some solid information.

Here are a few of the headlines from the report:

- The salary numbers themselves are astounding. The median U.S. salary for technical occupations was $27,849 for women and $40,546 for men. Our median (for system administrators) is $62,500!
- Finance, insurance, real estate, and entertainment industries are the highest paying for sysadmins, but such traditionally low-paying institutions as education are now closing the gap.
- Years of experience have a stronger relationship to higher compensation levels than a college degree. While any college degree helped get the $$, a computer degree did not get you more $$.
- 80% of all respondents expect to be systems managers five years from now!
- Sysadmins are people on the move! 40% of all respondents significantly changed what they did within the last year.
- There is no such thing as a simple site anymore. The average number of operating systems supported was 4.7. This means interoperability is key. Are you listening, Microsoft?
- You do not have to be a spreadsheet-wielding manager to earn the bucks. Redefining "career advancement" is on the agenda of many systems managers.

Working at a university, I normally see a few of the better students apply to graduate school. This year, none of my students is applying to graduate school. Rather, a sizable number of students (undergraduate and graduate) are delaying their schooling to enter the work force, to take advantage of opportunities before the bubble bursts. Systems managers are being pulled along with this tide of money. Competition for the smart, bright, motivated, and skilled person is intense. Toys are tossed in our paths to entice us to another "opportunity." It's industry, trolling for minds. Startups are not the sole cause of this vacuum. Long-standing industries are reengineering their computing infrastructure; all commerce is being channeled to the Web. It is not surprising that some of the highest-paid individuals run Web farms.

The complete report of the 1999 SAGE System Administrator Salary Profile is now available to anyone for the asking at *<http://www.usenix.org/sage/jobs/salary_survey/>*. I urge everyone to download a copy. It has much more information than could be presented here, including salary information by zip code, industry, and SAGE job categories. Read it, and let us know what you think. I look forward to active discussion on the *<sage-members>* mailing list!

The salary profile evolved from a spontaneous paper-based "sysadmin profile" at the LISA conferences. This year's results are based on our new Web-based questionnaire, which will continue to evolve each year. The 2000 Salary Profile will be refined and expanded to consider the international aspects of our membership and professional responsibilities. I feel the sample size of 2,314 is too small and would like to grow that number tremendously with the next iteration.

I'd like to take a moment to thank all the respondents. The SAGE Executive Committee hopes this information will improve your daily working life and give more form to the fine art of systems management!

Report on the SAGE System Administrator Profile Survey

As part of its ongoing effort to gain recognition and advancement for system administrators, SAGE annually conducts a System Administrator Profile Survey. This report is based on results of the 1999 profile that was administered at the LISA Conference and on the USENIX/SAGE Web site during November and December 1999.

The 2,314 respondents to the 1999 survey worked in 48 different countries. The majority (81.6%) worked in the U.S. The other most common countries were Canada (5.0%), Australia (3.8%), and the United Kingdom (1.9%). Most analyses of salary, bonuses, and total cash (total cash is salary, wages, bonuses, and other nondeferred cash payments) were based on only the U.S. respondents, because of the small sample sizes for other countries.

Most respondents were salaried workers (90.1%), working for a single employer (90.9%), male (87.3%), and had a bachelor's degree (46.0%) or some college (21.5%) as their highest level of education. Most worked with Solaris (77.7%), Windows NT (63.7%), Linux (55.9%), and/or Windows 95/98 (47.8%). On average, respondents worked with 4.7 operating systems. The majority did not supervise any subordinates (72.0%) and were not certified on any operating system (65.4%). They averaged 11.3 days of travel per year, 47.0 hours of work per week, 7.9 years of experience in the field, 2.8 different employers while in the field, and 34 years of age. Half had worked for their current employer two years or less. The majority (89.1%) indicated that system administration was their primary line of work. Over one-fourth of respondents were in the computer/software/Internet industry, over 15% worked in university or college education, and over 10% were in consulting/business services.

For U.S. system administrators, mean salary was $64,271 and the median was $62,500; the mean total cash was $70,565 and the median was $65,200; the mean bonus was $3,464 and the median was zero. Mean and median salaries were lower in other countries and areas of the world than in the U.S., particularly in Eastern Europe/Western Asia. Mean salaries were over $75,000 for two New York City zip codes (10 and 11), one North Carolina zip (28), and the Sacramento and San Francisco area (94 and 95) zip codes.

For those in the U.S., the average 1999 pay increase from the same employer for the same job was 7.9%; from the same employer for a promotion, 14.9%; and from changing employers, 23.3%. Over 61% had received an increase from the same employer for the same job. More than one in six had received an increase from changing employers, and more than one in ten had received a promotional increase.

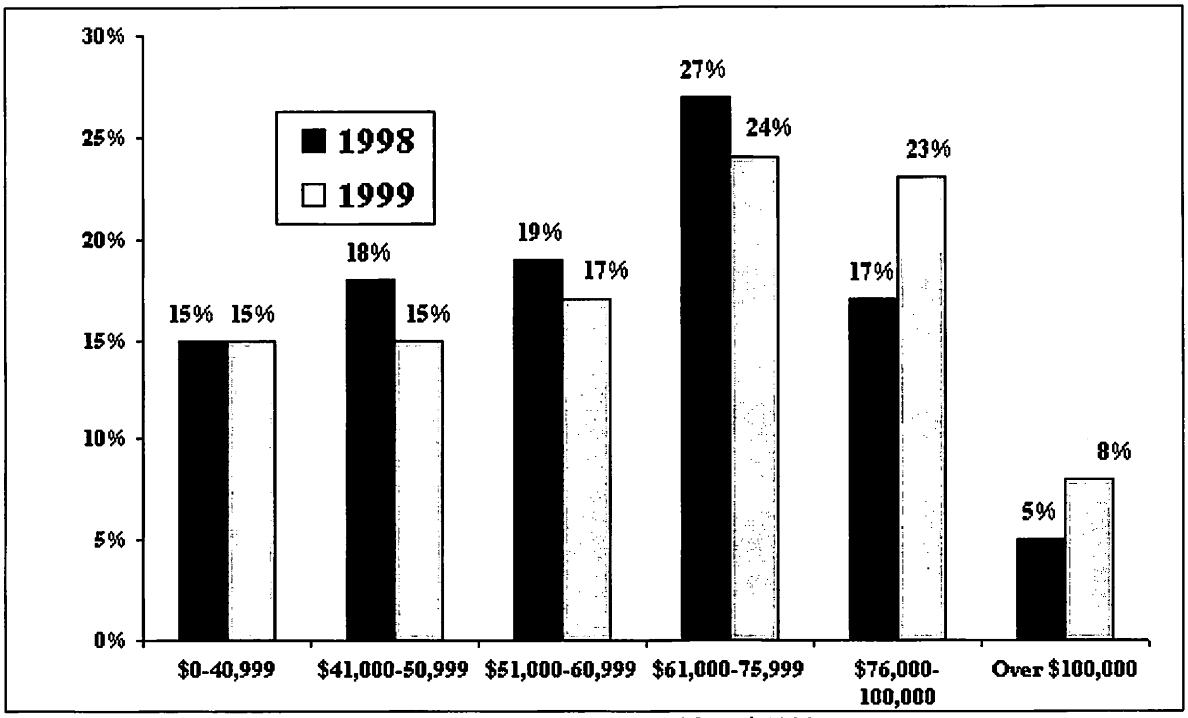


Figure 1: Income ranges, 1998 and 1999

Figure 1 shows the percentage of respondents for each of the two years who had income in the ranges shown. The percentage with salary less than $41,000 was the same in 1999 (15%) as in 1998 (15%), because the sample included a higher proportion of respondents from outside the US (29.0% in 1999 compared with 14.4% in 1998), where salaries are lower, on average. Percentages in the middle salary ranges decreased, while the percentage making

$76,000 or more increased in 1999 (31%) compared to 1998 (22%), which demonstrates the trend of increasing salaries for system administrators.

Over half (52.1%) of the system administrators in the total sample did not receive a bonus. For those who did, Figure 2 shows the percentage of the sample who received each type of bonus. Over 22% received a bonus for individual performance; 17.1% received a bonus for organizational performance, 10.1% for group or unit performance, and 6.7% for a special project. The median bonus for individual and organizational performance was $3,000, for group/unit performance it was $2,000, and for a special project it was $1,500. Of the 21 respondents who reported a bonus related to stock, the median was $27,000.

Table 1. Total Cash in 1999 by Type of Primary Job (U.S. System Administrators)

Job Type	# of Respondents	Mean Total Cash in U.S. $
Salaried	1,470	$68,742
Contractor	13	$78,346
Independent, self-employed consultant	24	$106,746

Table 2. Total Cash in 1999 by SAGE Job Description Level (U.S. System Administrators)

Job Level	# of Respondents	Mean Total Cash in U.S. $
Level 1	43	$50,766
Level 2	181	$57,806
Level 3	851	$66,578
Level 4	554	$81,021
Mean for All Levels	1,629	$70,098

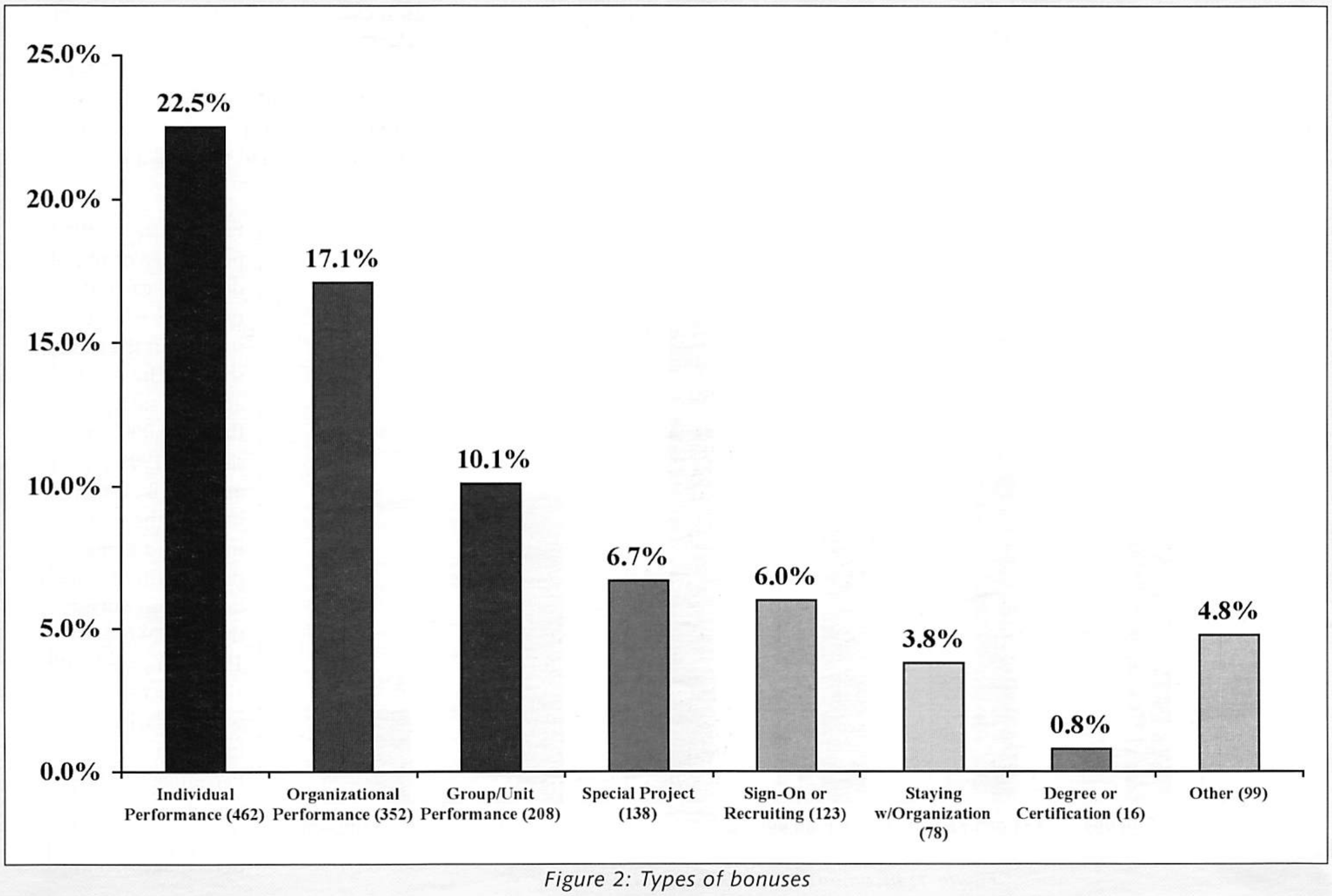


Figure 2: Types of bonuses

Eight percent of the system administrators in the sample said they were contractors, and 1.9% said they were independent, self-employed consultants. Mean salary was lower for salaried ($62,152) than contractor ($77,307) or consultant ($96,121) system administrators in the U.S. The same pattern was found for mean total cash, as shown in Table 1.

Table 3. 1999 Salary by Level of Education (U.S. system administrators)

Level of Education	# of Respondents	Mean Salary in U.S. $
Certificate	6	$46,050
High School	44	$58,055
Some Technical School	77	$60,533
Some College	383	$61,076
Associate's Degree	123	$61,585
Bachelor's Degree	770	$64,641
Master's Degree	237	$67,998
Ph.D.	20	$72,160

Mean salary for those who "assist in the administration of a systems facility" and "perform routine tasks under the direct supervision of a more experienced system administrator" (Level 1 of the four SAGE Job Description Levels) was $47,847. For those at Level 2, who "work under the general supervision of a computer system manager, carrying out more complex tasks with some independence," the mean was $54,765. For those at Level 3, who "initiate some new responsibilities and help to plan for the future of the facility," "manage the work of novice system administrators," and "evaluate and/or recommend purchases," the mean was $61,703. And for those at Level 4, who "design and manage the computing infrastructure," "work under general direction from senior management," and "establish or recommend policies on system use and services," the mean was $71,725. The same pattern of relationship between increasing level and increasing pay is seen for total cash in the Table 2.

In the U.S., supervision of subordinates was related to the following average salaries: no subordinates, $62,448; 1 to 2, $63,297; 3 to 5, $67,779; 6 to 10, $70,044; and 11 or more, $78,168.

Table 4. 1999 Salary and Total Cash by Sex (U.S. System Administrators)

Sex of Respondent	# of Respondents	Mean Salary in U.S. $	Mean Total Cash in U.S. $
Male	1,435	$64,883	$71,214
Female	225	$57,777	$62,863

For U.S. system administrators, salary also tended to increase with hours worked (e.g., $41,848 average for 0–19 hours; $54,449 for 20–34 hours; $62,900 for 41–45 hours; $66,156 for 51–60 hours).

For U.S. system administrator respondents, 15.4% reported receiving overtime pay, 0.1% shift pay, and 15.9% on-call/pager pay. Overtime pay for U.S. system administrators was most prevalent in aerospace, government, and the military. Shift pay was most prevalent in the aerospace industry and federal government (nonmilitary). Over one-quarter of U.S. respondents in health care, state/local government, and transportation organizations said they get paid for being on call or wearing a pager.

The number of operating systems supported was not related to the amount of pay, but those who worked with Solaris, BSDI, and HP-UX tended to have significantly higher salaries, while those who worked with FreeBSD, MacOS (non-UNIX), OpenBSD, or DOS/Win 3.1 tended to have lower salaries, on average. About a third of the sample (34.5%) had a certification for an operating system. Those with certifications for HP-UX, Solaris, and SunOS tended to earn higher salaries or more total cash. Other operating systems and certifications, including Linux and Windows NT, did not have a statistically significant relationship with the amount of salary or total cash earned.

As shown in Table 3, U.S. system administrators with high school as their highest level of education averaged $58,055 in salary, while those with increasingly higher levels of education made increasingly higher average salaries. The number of certifications one had earned was related to salary in the U.S.: those with no certifications averaged $62,992 in salary; those with one certification averaged $64,116; and those with six or more averaged $71,322. Members of SAGE tended to receive higher salaries ($65,818, on average) and more total cash ($73,990, on average) than those who were not members of a professional organization (average $61,637 in salary and $65,226 in total cash). As shown in Table 4, male system administrators reported higher average salary and total cash than the female system administrator respondents from the U.S. Regression equations including variables such as education, years of experience, and job level along with sex indicated that females make lower salaries but higher bonuses, on average, and do not have significantly lower total cash when the other variables are taken into account.

The number of years of experience as a system administrator (or in similar work) was positively related to higher salaries and total cash (see Table 5), but the number of years with one's current employer was not. Those with two years or less of experience averaged less than $50,000 salary; those with three to ten years averaged between $50,000 and $60,000; those with 11 to 14 years averaged between $70,000 and $75,000; and those with over 15 years averaged more than $76,000. Age was also positively related to compensation, although the relationship was not as strong as that between years of experience and compensation.

Table 5. Salary by Years of Experience (U.S. System Administrators)

Years of Experience	# of Respondents	Mean Salary in U.S. $
1 or less	58	$38,163
2	90	$46,178
3	138	$54,288
4	159	$55,508
5	204	$60,246
6	145	$63,643
7-8	224	$67,814
9-10	196	$67,510
11-12	142	$72,167
13-14	83	$74,749
15-16	109	$76,958
17-20	73	$76,934
21-25	29	$77,950
26 or more	8	$77,780

Table 6 illustrates how U.S. system administrators' compensation was positively related to the number of employers they had had during their career. This was not because those who had had more employers also had more experience; nor was it due to the fact that those with more employers were more likely to be contractors or consultants. In addition, those who worked for more than one employer at the time of the survey (i.e., more than one at the same time) averaged about $5,000 more salary ($68,573 vs. $63,463) and about $6,000 more total cash ($75,622 vs. $69,589) than those with one employer.

Compensation varied considerably by industry for U.S. system administrators. The median salary was lowest in state or local government ($49,000) and colleges or universities ($50,000) and highest in finance, insurance, and real estate ($71,500) and entertainment ($80,000). These were also the industries with the lowest and highest median total cash, respectively. State/local government, entertainment, and utilities were among the most generous in providing fully paid insurance, although this varied somewhat by type of insurance.

The numbers of computers supported, users supported, and system administrators employed where one worked were not related to pay levels. The number of computers or users per administrator in one's facility was also not related to compensation.

The number of employees in one's organization, worldwide, was significantly related to salary (see Table 7). Salary tended to go up with the number of employees. The mean number of paid holidays, vacation days, and training

Table 6. 1999 Salary and Total Cash by Number of Employers (U.S. system administrators)

# of Employers	# of Respondents	Mean Salary in U.S. $	Mean Total Cash in U.S. $
1	291	$55,338	$58,460
2	445	$58,925	$62,537
3	419	$65,002	$68,901
4	243	$70,397	$77,853
5	123	$74,876	$83,240
6	55	$71,836	$92,523
7	26	$78,595	$137,815
8 or more	26	$87,667	$108,536

days, and the percentage with a retirement plan, childcare assistance, or tuition assistance also tended to increase with the number of employees in an organization. The percentage who could telecommute or use flextime was lower for those in organizations with 51 to 2,500 employees than in either smaller or larger organizations.

U.S. system administrators averaged 15.4 days of paid vacation, 11.1 days of paid sick leave, 8.8 paid holidays, and 8.3 days of paid training per year. Paid time off tended to be highest in government, college/university, military, and not-for-profit organizations. Paid training days were highest in the military. The only industry for which U.S. respondents reported a higher percentage of defined benefit (i.e., pension) than defined contribution (e.g., 401k, 403b) retirement plans was state/local government.

Table 7. 1999 Salary by Organization Size (U.S. system administrators)

Number of Employees	Percent of Responses	Mean Salary in U.S. $
1	0.5%	$58,788
2–10	1.8%	$48,032
11–50	7.1%	$61,356
51–500	25.9%	$64,174
501–10,000	37.2%	$62,478
10,001 or more	27.5%	$67,941

The majority (71.9%) of U.S. respondents said "yes," their organization does have difficulty filling all of the system administrator positions it would like to fill; 17.9% said "no" and 10.2% said "not sure" to this question. The industries with percentages above 75% were retail and wholesale trade, consulting and business services, federal government–nonmilitary, aerospace, college/university education, and advertising/public relations/communication/marketing. Geographical areas with the highest percentages were San Diego, the Research Triangle of North Carolina, and the San Francisco area.

Over half of the respondents said the factors that would be most important in making them think seriously about switching jobs were pay (83.1%), location (76.4%), and benefits (60.0%). Over one-third marked organizational stability (38.9%), hours (38.6%), and organizational reputation (38.3%). (Respondents could mark more than one category.) The majority of respondents (80.1%) said they expect still to be a system administrator in five years.

Respondents were asked to note any special benefits or working conditions they particularly liked and the most problematic or bothersome aspects of their jobs. The following are ranked according to the number of responses (e.g., #1 had the most respondents).

Regression analysis was used to determine which of the various job, organizational, and personal background characteristics on the survey were most highly related to compensation. Over half (52.0%) of the variance in salary, 23.4% of the variance in total cash, and 8% of the variance in bonuses were accounted for by the equations. Thus, there were more systematic relationships between survey topics and salary than between survey topics and total cash or bonuses. The most significant factors associated with salary were being a consultant or contractor, location, industry, operating systems used, job level, education, number of employers, and experience.

More detailed versions of these data and analyses are available from USENIX/SAGE upon request.

Benefits/Working Conditions That Were Liked

1. Flexible work schedules
2. Jobs that provide challenge, autonomy, variety, learning
3. Organizational cultures that are university-like, noncorporate, friendly, or nonbureaucratic
4. Good managers, co-workers, and users
5. Equipment for home offices provided by employers
6. Casual dress or lack of a dress code
7. Free beverages and/or food
8. Telecommuting
9. Working with new technology and good resources
10. Good pay, special awards, stock/options, or paid overtime

Most Problematic/Bothersome Aspects of Jobs

1. Poor management
2. Long hours, heavy workloads, or being on call
3. Office politics and bureaucracy
4. Low pay or lack of pay for overtime or on-call time
5. Poor resources, low budgets, or lack of help desk support
6. Routine, unchallenging, menial, or administrative tasks
7. Users not using resources to solve problems themselves
8. Lack of training, career development, and career paths
9. Understaffing, recruiting, and retention problems
10. Long commutes, extensive travel, or travel without notice

Announcement and Call for Papers **USENIX**

3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)

http://www.usenix.org/events/usits01

March 26-28, 2001

Cathedral Hill Hotel, San Francisco, California, USA

Important Dates

Paper submissions due: *September 18, 2000*
Notification of acceptance: *November 14, 2000*
Revised papers due for shepherding: *Dec. 15, 2000*
Camera-ready final papers due: *January 23, 2001*

Conference Organizers

Program Chair
Tom Anderson, *University of Washington*
Program Committee
Mary Baker, *Stanford University*
Hari Balakrishnan, *MIT*
Eric Brewer, *UC–Berkeley*
Steve Gribble, *University of Washington*
Steve McCanne, *Fast Forward*
Vivek Pai, *Princeton University*
Brian Pinkerton, *Excite*
Bill Weihl, *Akamai Technologies*
Willy Zwaenepoel, *Rice University*

Overview

The goal of this symposium is to bring together a diverse collection of engineers and researchers to help define the future of Internet applications, technologies, and systems. We actively encourage participation by experts in distributed systems, networking, performance measurement, security, user interfaces, artificial intelligence, operating systems, and system administration. We believe future Internet innovations will emerge from a fusion of the best ideas from a spectrum of areas.

This will be a 3 day symposium, featuring refereed paper presentations, invited talks, Works-in-Progress presentations, demos, panel discussions, and Birds-of-a-Feather sessions.

Technical Sessions

The Internet continues to evolve in interesting and unexpected ways. Electronic commerce, mobility, streaming media, and other developments are driving the creation of new applications, protocols, security models, and systems. Recent application-level changes, such as XML, may dramatically improve the functionality of the Web, and an increasing number of consumer devices are being built to be Internet ready. What's next?

USITS emphasizes both innovative research and quantified experience in Internet applications, technologies, and systems. We seek papers describing original work concerning the design, implementation, and application of Internet technologies. Besides mature work, we also encourage papers outlining a compelling distant vision of the future, exceptionally promising prototypes, or enlightening negative results. Case studies and experience papers are particularly welcome.

Topics

Topics of interest include, but are not limited to:

- Caching and co-location services
- Content distribution
- Distributed systems
- Electronic commerce
- Information indexing/retrieval
- Internet agents
- Internet programming tools
- Novel system architectures
- Performance
- Resource discovery/management
- Security
- Server OS and I/O
- Ubiquitous computing
- User interfaces

see over →

Best Paper Awards

Awards will be given for the best paper and best student paper at the symposium.

What to Submit

Submissions should be full papers, 10-12 single-spaced 8.5" x 11" pages, including figures, tables, and references, using an 11 point or larger font. A good paper will demonstrate that the authors:

- are attacking a significant problem,
- have devised an interesting, compelling solution,
- have demonstrated the practicality and benefits of the solution,
- have drawn appropriate conclusions,
- have clearly described what they have done, and
- have clearly articulated the advances beyond previous work.

Submissions will be judged on originality, significance, interest, clarity, relevance, and correctness. Please read the detailed author guidelines on the symposium Web site for more information.

Submissions are due on September 18, 2000. Accepted papers will be shepherded through an editorial review process by a member of the program committee. Based on initial feedback from the program committee, each author will submit an editorial revision of their paper to their program committee shepherd by December 15, 2000. The program committee member will review the paper and give the author additional comments. The author will then produce a final camera-ready paper by January 23, 2001, for the symposium proceedings.

USITS, like most conferences and journals, requires that papers not be submitted simultaneously to any other conference or publication, that submissions not be previously published, and that accepted papers not be subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

How to Submit

Authors are required to submit full papers by September 18, 2000. All submissions for USITS '01 will be electronic, in PDF, PostScript, or ASCII. A Web form will be available by May on the symposium Web site.

Authors will be notified of receipt of submission via e-mail. If you do not receive notification, contact: *usitschair@usenix.org*.

Demonstrations

The symposium will include a session where participants can present and demonstrate their work in an informal setting. To schedule a demo of your work at the symposium, please email *usitsdemos@usenix.org*.

Work-in-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not ready to be published? Work-in-progress reports are for you! Work-in-progress reports, scheduled during the technical sessions, introduce new or ongoing work. The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentations of student work. To submit a work-in-progress, please send a proposal, one page or less, by March 9, 2001, to the work-in-progress coordinator at *usitswips@usenix.org*.

Birds-of-a-Feather Sessions

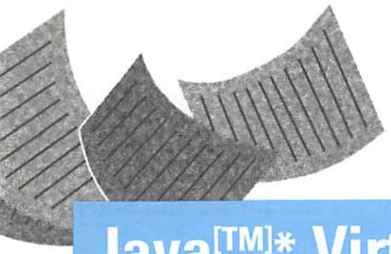
Birds-of-a-Feather sessions (BoFs) are very informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evenings. BoFs may be scheduled in advance by phoning the Conference Office at +1.949.588.8649 or via email to *conference@usenix.org*. BoFs may also be scheduled at the symposium.

Registration Materials

Materials containing all details of the technical program, registration fees and forms, and hotel information will be available by December, 2000 on the symposium Web site. If you wish to receive the registration materials in print, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

4/14/00



Announcement and Call for Papers **USENIX**

Java[TM]* Virtual Machine Research and Technology Symposium (JVM '01)

http://www.usenix.org/events/jvm01

April 23-24, 2001 **Monterey, California, USA**

Important Dates

Paper submissions due: *November 1, 2000*
Notification of acceptance: *December 15, 2000*
Camera-ready final papers due: *February 27, 2001*
Symposium begins: *April 23, 2001*

Conference Organizers

Program Chair
Saul Wold, *Sun Microsystems*
Program Committee
Tony Cocchi, *IBM*
Urs Hoelzle, *University of California, Santa Barbara*
Juergen Kreileder, *The Blackdown Project*
Tim Lindholm, *Sun Microsystems*
Additional members to be announced.

Overview

For the first Java Virtual Machine Research and Technology Symposium, we invite the submission of quality papers describing research or experiences with the Java Virtual Machine. Research papers should describe original work that offers significant contributions to the state of JVMs. Experience papers should describe general insights gained from porting, integrating, or tuning JVMs. Submitted papers should make substantial contributions to the field and should be useful to researchers and practitioners alike.

This symposium will have 2 days of technical sessions, made up of a Keynote Address, Refereed Papers, and a Work in Progress session.

JVM '01 emphasizes research and advanced engineering aspects of the Java Virtual Machine, focusing on experimental research. The Symposium Proceedings, containing all refereed papers, will be distributed to attendees and, following the symposium, will be available online to USENIX members and for purchase.

Awards will be given at the symposium for the best paper and for the best paper that is primarily the work of a student.

Topics

Relevant topics for JVM '01 include, but are not limited to the following:

- Alternate VM implementation
- Hardware implementation
- JITs and Execution Engines
- Security and VM issues
- Garbage Collection techniques
- Small JVM (á la JavaCard)
- Large JVM and Server Issues (scalability and availability)
- Porting Experience Improvements
- Performance Issues and Tuning Techniques

Questions about the relevance of a topic may be addressed to the Program Chair at *jvm01chair@usenix.org*

What to Submit

Submissions should be full papers, 10 to 14 pages (around 5,000-6,000 words) in length. Papers that are too long or are late will be rejected. All submissions will be judged on originality, significance, relevance, correctness, and clarity. Each submission must include the paper title, the contact author, email and regular addresses, and a phone number. For more information, please read the detailed author guidelines, located on the symposium Web site.

The JVM symposium, like most conferences and journals, requires that papers not be submitted simulta-

*Java is a Trademark of Sun Microsystems, Inc.

see over →

neously to any other conference or publication, that submissions not be previously published, and that accepted papers not be subsequently published elsewhere for a year from date of acceptance by USENIX. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

How to Submit

Web-based electronic submission will be expected. Submissions should be in Postscript that is interpretable by Ghostscript or in PDF that is interpretable by Acroread, and should be printable on US Letter sized paper. A Web form for submissions will be available in April on the symposium Web site at: *http://www.usenix.org/events/jvm01.* All submissions will be acknowledged.

Submitters for whom web submission is a hardship should contact the Program Chairs for alternative means of submission at *jvm01chair@usenix.org.*

Work In Progress Reports

For this JVM Symposium we will include a session on "work in progress" (WIP) to introduce new ideas to the community and solicit early feedback. We are particularly interested in the presentation of student work and bleeding edge VM implementation and intergration in both software and hardware. WIP abstracts will be lightly screened to facilitate focused discussions during the session. The submission process for WIP abstracts will begin in January 2001. Full submission information will be available at the symposium Web site.

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are very informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by phoning the Conference Office at +1 (949) 588-8649 or via email to *conference@usenix.org.* BoFs may also be scheduled at the symposium.

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available by January 2001 at the symposium Web site. If you wish to receive the registration materials in print, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

USENIX ASSOCIATION Conference Proceedings and

Publications Order Form

USENIX

Publications Order Form

Quantity	Title	Location/Date	Price: Member/List	Overseas Postage	Total
General Technical Conferences					
______	1999 Annual	Monterey, CA	$32/40	18.	______
______	1999 FREENIX Track - Annual	Monterey, CA	$22/30	18.	______
______	1998 Annual	New Orleans, LA	$32/40	18.	______
______	1997 Annual	Anaheim, CA	$32/40	18.	______
______	1996 Annual	San Diego, CA	$32/40	18.	______
______	1995 Annual	New Orleans, LA	$30/39	20.	______
______	Summer 1994	Boston, MA	$25/33	20.	______
______	Winter 1994	San Francisco, CA	$30/39	20.	______
Systems Administration (LISA and LISA-NT) Conferences					
______	Systems Administration (LISA XIII)	November 1999	$32/40	12.	______
______	2nd Large Installation System Administration of Windows NT	July 1999	$18/24	10.	______
______	Systems Administration (LISA XII)	December 1998	$32/40	12.	______
______	Large Installation System Administration of Windows NT	August 1998	$18/24	12.	______
______	Systems Administration (LISA XI)	October 1997	$30/38	12.	______
______	Systems Administration (LISA X)	September 1996	$30/38	20.	______
______	Systems Administration (LISA IX)	September 1995	$30/38	20.	______
______	Systems Administration (LISA VIII)	September 1994	$22/29	20.	______
Networking					
______	Network Administration (NETA)	April 1999	$18/24	11.	______
______	Intrusion Detection and Network Monitoring (ID)	April 1999	$23/30	11.	______
______	High-Speed Networking	August 1994	$15/20	9.	______
Security					
______	Security VIII	August 1999	$27/35	18.	______
______	Security VII	January 1998	$27/35	18.	______
______	Security VI	June 1996	$27/35	18.	______
______	Security V	June 1995	$27/35	18.	______
Operating Systems					
______	Operating System Design & Implementation	February 1999	$23/30	11.	______
______	Operating System Design & Implementation	October 1996	$20/27	11.	______
______	Operating System Design & Implementation	November 1994	$20/27	11.	______
Programming Languages					
______	2nd Conference on Domain-Specific Languages	October 1999	$20/24	11.	______
______	Conference on Domain-Specific Languages	October 1997	$20/24	12.	______
______	7th Tcl/Tk Workshop	February 2000	$24/32	12.	______
______	6th Tcl/Tk Workshop	September 1998	$22/28	12.	______
______	5th Tcl/Tk Workshop	July 1997	$22/28	12.	______
______	4th Tcl/Tk Workshop	July 1996	$22/28	12.	______
______	3rd Tcl/Tk Workshop	July 1995	$29/34	20.	______
______	Conf. on Object-Oriented Technologies & Systems V	May 1999	$24/32	12.	______
______	Conf. on Object-Oriented Technologies & Systems IV	April 1998	$22/32	12.	______
______	Conf. on Object-Oriented Technologies & Systems III	June 1997	$20/30	12.	______
______	Conf. on Object-Oriented Technologies & Systems II	June 1996	$20/30	12.	______
______	Conf. on Object-Oriented Technologies & Systems I	June 1995	$18/24	9.	______
______	Very High Level Languages	October 1994	$23/30	10.	______
______	C++ Conference	April 1994	$24/28	20.	______

→

Quantity			Price: Member/List	Overseas Postage	Total
Other Symposia & Workshops					
______	Workshop on Smartcard Technology	May 1999	$22/28	12.	______
______	Workshop on Embedded Systems	March 1999	$20/28	12.	______
______	3rd USENIX Windows NT Symposium	July 1999	$18/24	10.	______
______	2nd USENIX Windows NT Symposium	August 1998	$18/24	12.	______
______	USENIX Windows NT Workshop	August 1997	$18/24	12.	______
______	2nd Symposium on Internet Technologies & Systems	October 1999	$24/30	12.	______
______	Symposium on Internet Technologies & Systems	December 1997	$20/26	12.	______
______	Electronic Commerce Workshop	August/Sept. 1998	$20/26	12.	______
______	Electronic Commerce Workshop	November 1996	$20/26	11.	______
______	Electronic Commerce Workshop	July 1995	$20/26	10.	______
______	Mobile and Location Independent Computing II	April 1995	$18/24	9.	______
______	UNIX Applications Development	April 1994	$15/20	9.	______

Discounts are available for bulk orders of 10 or more of the same proceeding.

USENIX CD-ROM

______	LISA '99	November 1999	$65/90	3.50	______
______	Monterey '99	June 1999	$65/90	3.50	______
______	New Orleans '98	June 1998	$65/90	3.50	______
______	Anaheim '97	January 1997	$65/90	3.50	______

SAGE: Short Topics in System Administration Series

______	#1: Job Descriptions for System Administrators, 2nd ed	$5/7.50	3.50	______
______	#2: A Guide to Developing Computing Policy Documents	$5/7.50	3.50	______
______	#3: System Security: A Management Perspective	$5/7.50	3.50	______
______	#4: Educating and Training System Administrators: A Survey	$5/7.50	3.50	______
______	#5: Hiring System Administrators	$10/12	3.50	______

Note:
The member price also applies to members of EurOpen National Groups, AUUG, and JUS.

Reprints:
Reprints of individual papers from all proceedings are available for $5.00 each. (To get a complete listing of conference papers, please refer to our Online Library Index, available through the WWW at <http: //www.usenix.org>.

Payment Options

❑ Check enclosed payable to USENIX Association. ❑ Purchase order enclosed ❑ Visa ❑ MasterCard ❑ American Express

Account # ______________ Expiration Date ________ Signature ______________

Outside the USA? Please make your payment in US currency via one of the following:

❑ Check – issued by a local branch of a US bank ❑ Charge (Visa, MasterCard or equivalent) ❑ International postal money order

SHIP TO: ______________________

____

____

____

Shipping Information: Please allow 2-3 weeks for delivery. Shipping fees for domestic and Canadian orders are included. Please add postage fee for overseas orders (shipped via air printed matter).

Total Price of Publications	______
Calif. res. add sales tax: (8.25%)	______
Overseas Postage	______
TOTAL ENCLOSED $	______

If you are a member, please include your name and membership # to receive member price: ______________

❑ If you are not a member and wish to receive our membership information packet, please check this box.

Please return this order form and your payment to:
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710 Fax: 510/548-5738 Phone: 510/528-8649 *<office@usenix.org>* *<http://www.usenix.org>*

Rev. 5/03/00 mc

CONNECT WITH USENIX & SAGE

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITES

http://www.usenix.org

http://www.sage.org

EMAIL

login@usenix.org

COMMENTS? SUGGESTIONS?

Email: login@usenix.org

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send address changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

I36935